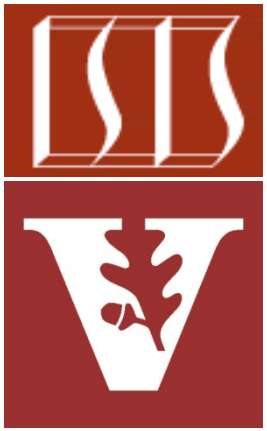
Introduction:
Overview of Patterns & Frameworks
(Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



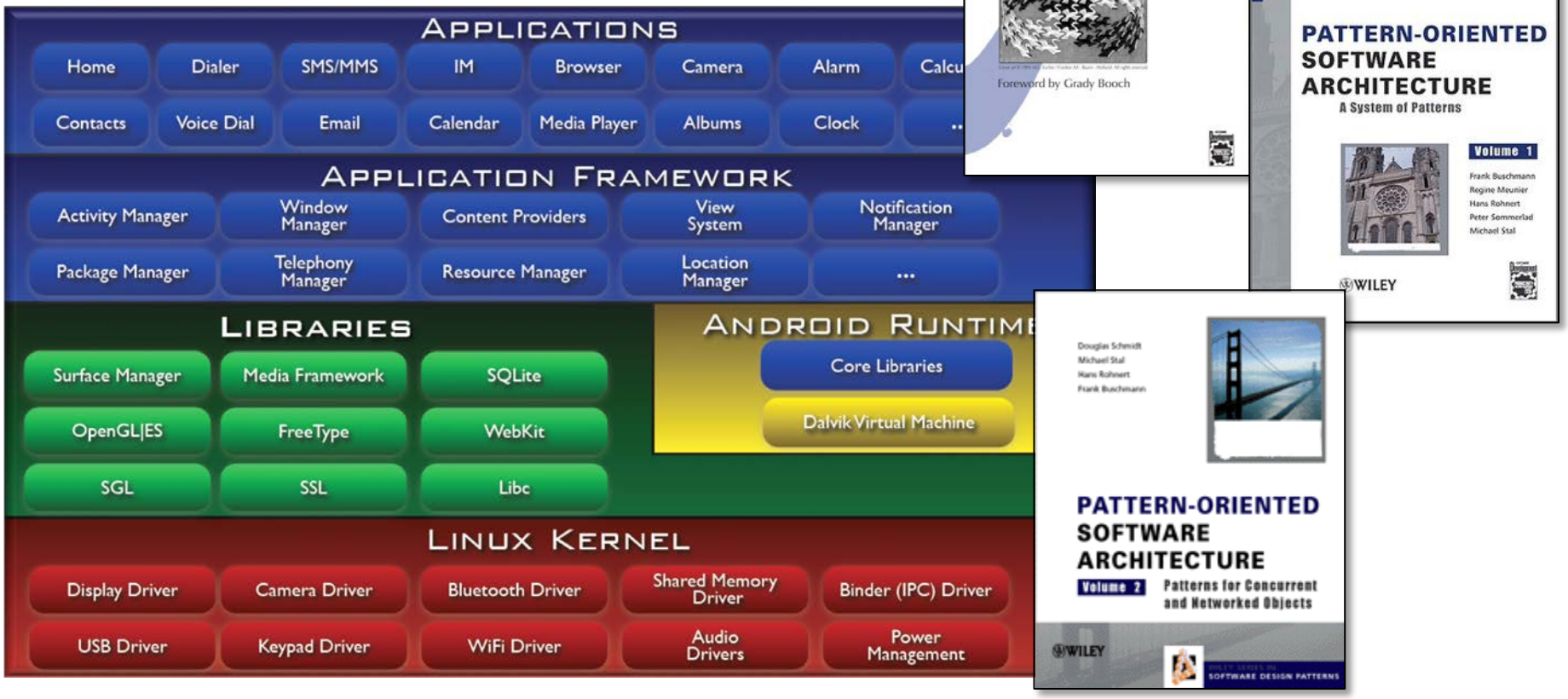
Professor of Computer Science

Institute for Software
Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

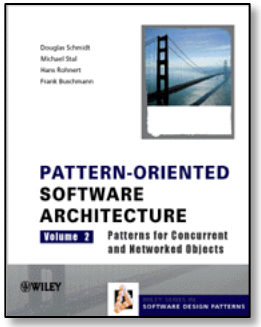
Learning Objectives in this Part of the Module

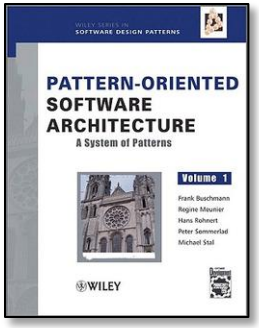
- Understand how *patterns* & *frameworks* help improve the structure & functionality of Android's concurrency & communication middleware used by Applications & Services

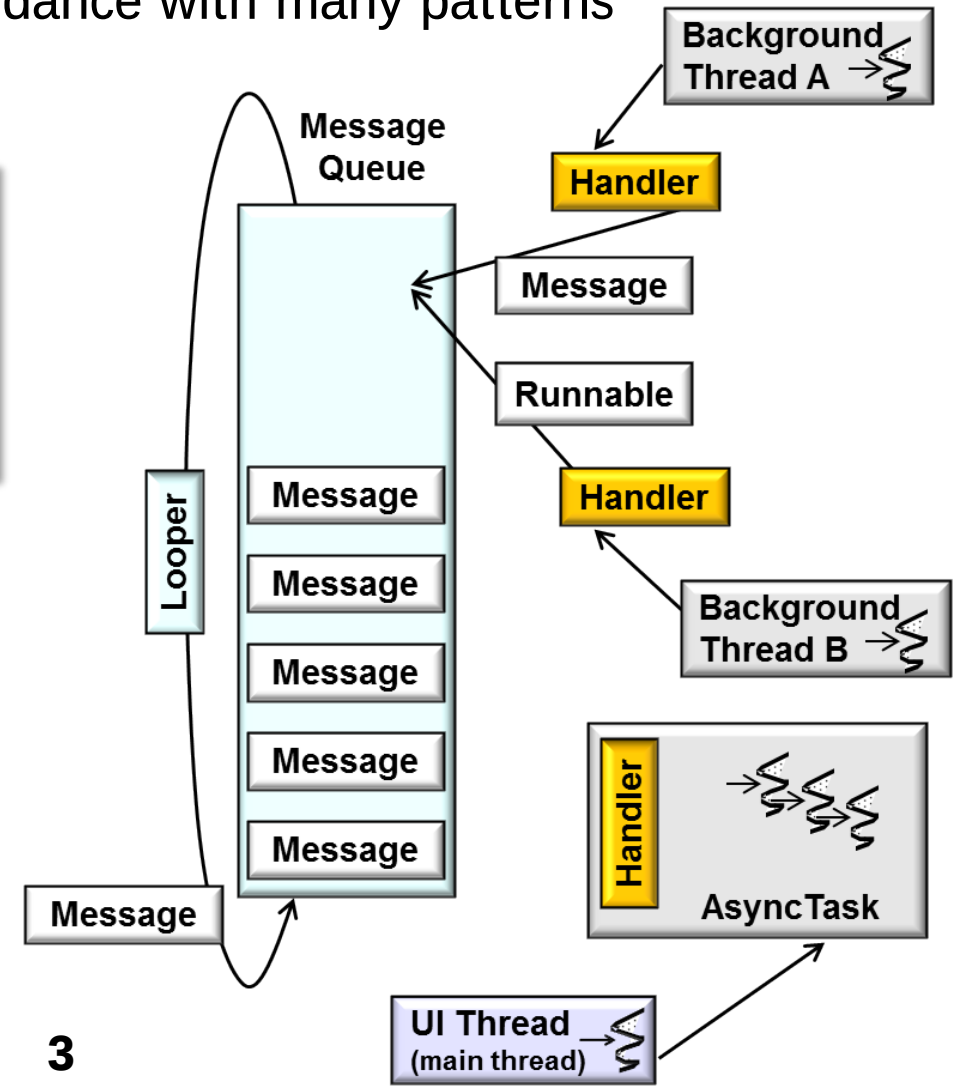


Patterns Applied in Android Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns

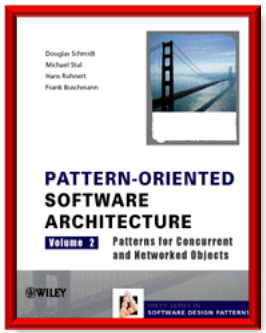


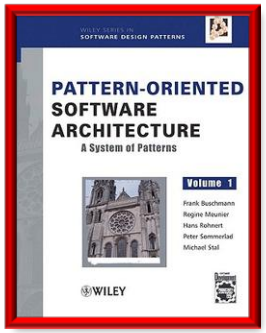


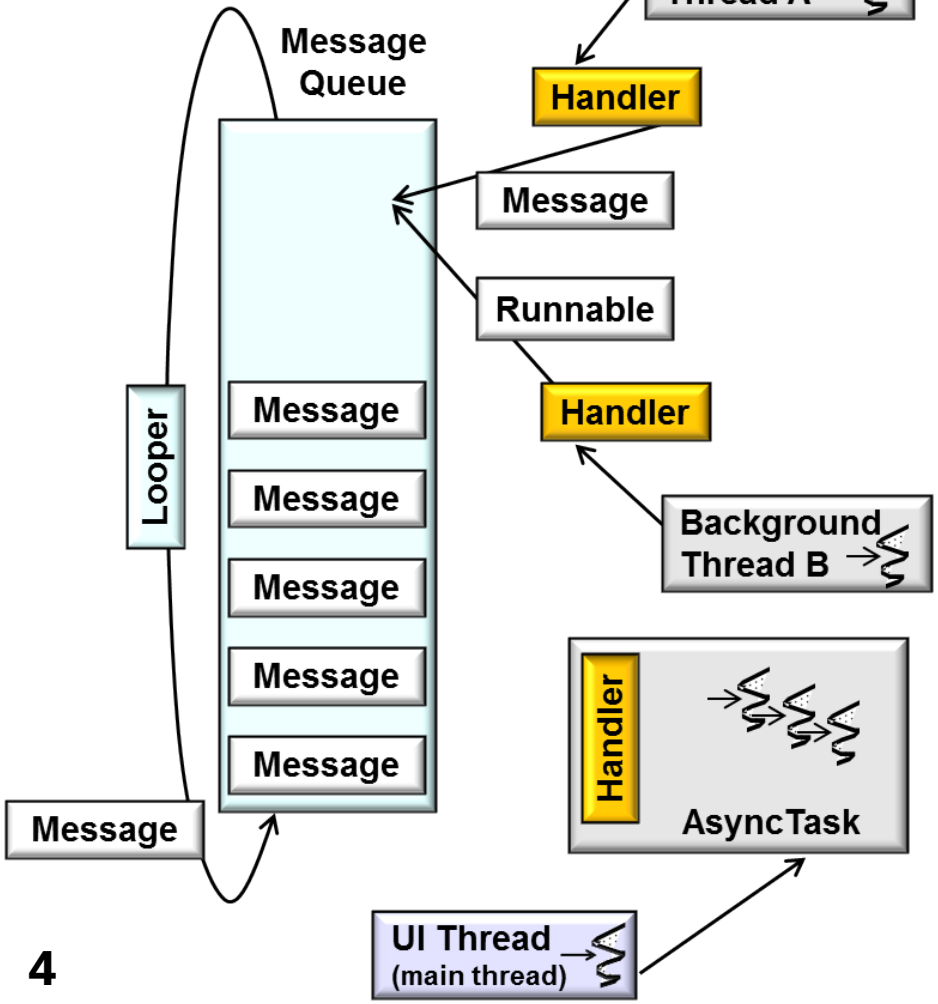


Patterns Applied in Android Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns

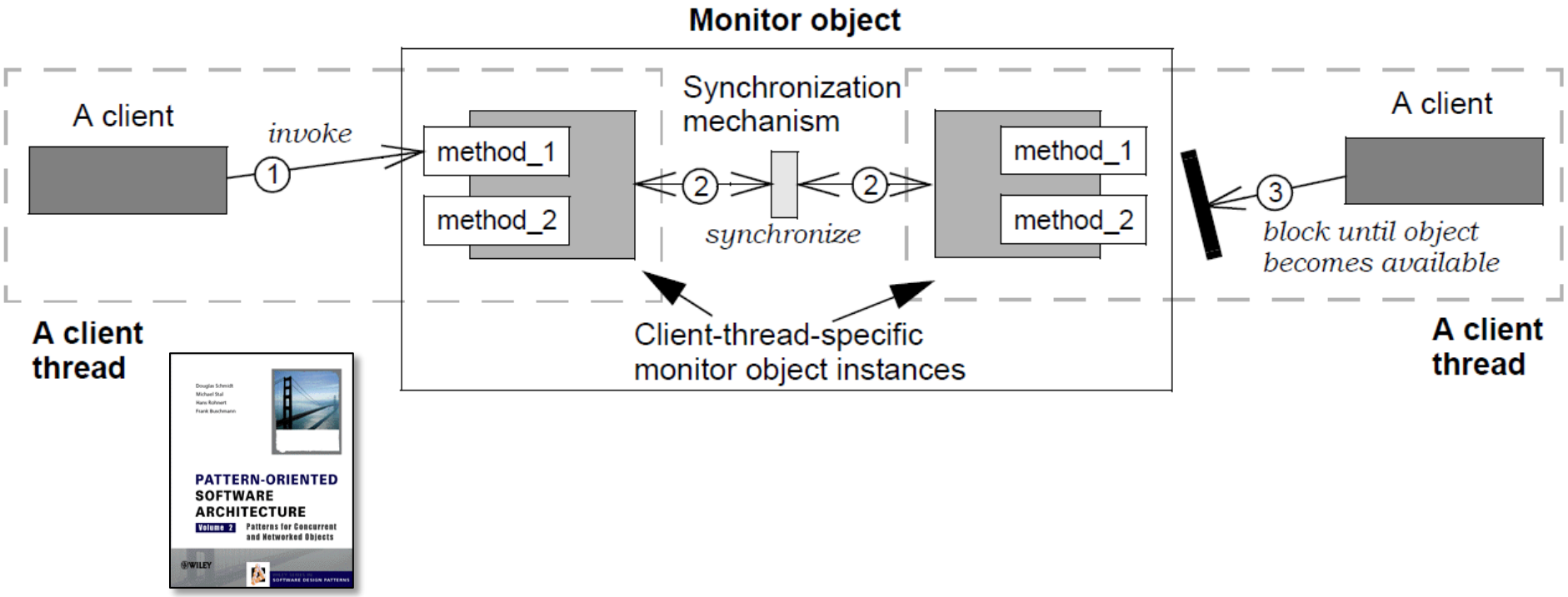






Patterns Applied in Android Frameworks

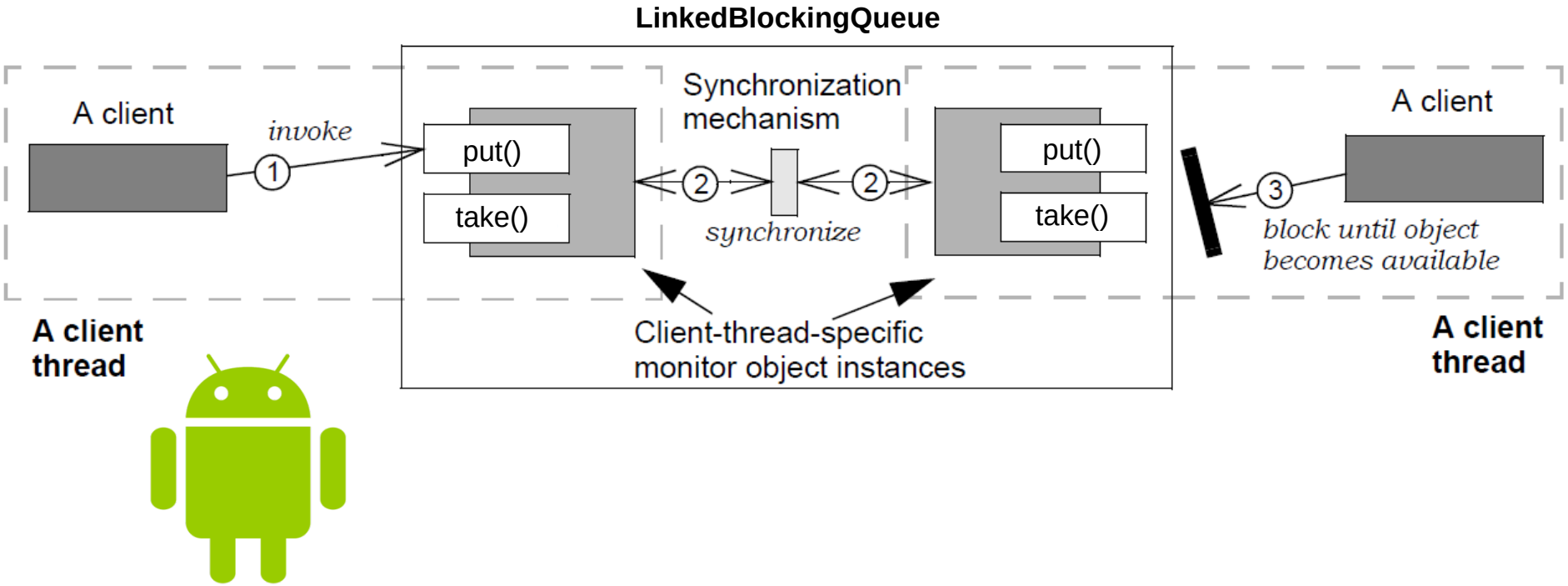
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Monitor Object* – Synchronizes concurrent method execution to ensure only one method at a time runs within an object & allows an object's methods to cooperatively schedule their execution sequences



www.dre.vanderbilt.edu/~schmidt/PDF/monitor.pdf has more info

Patterns Applied in Android Frameworks

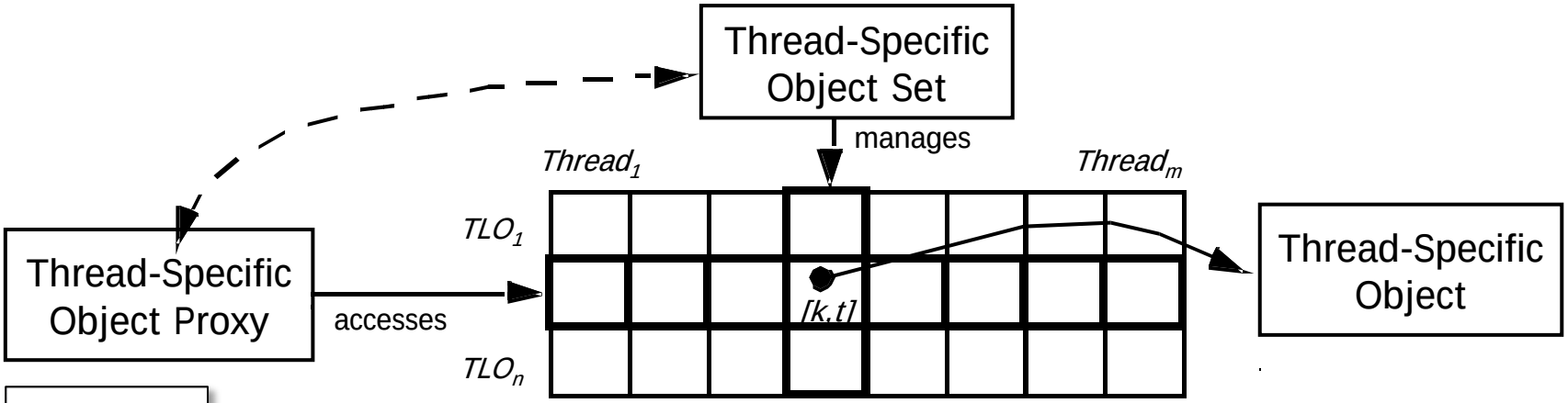
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Monitor Object* – Synchronizes concurrent method execution to ensure only one method at a time runs within an object & allows an object's methods to cooperatively schedule their execution sequences

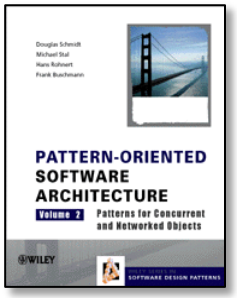


See upcoming parts on "The *Monitor Object* Pattern"

Patterns Applied in Android Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Thread-Specific Storage* – allows multiple threads to use one 'logically global' access point to retrieve an object that is local to a thread, without incurring locking overhead on each object access

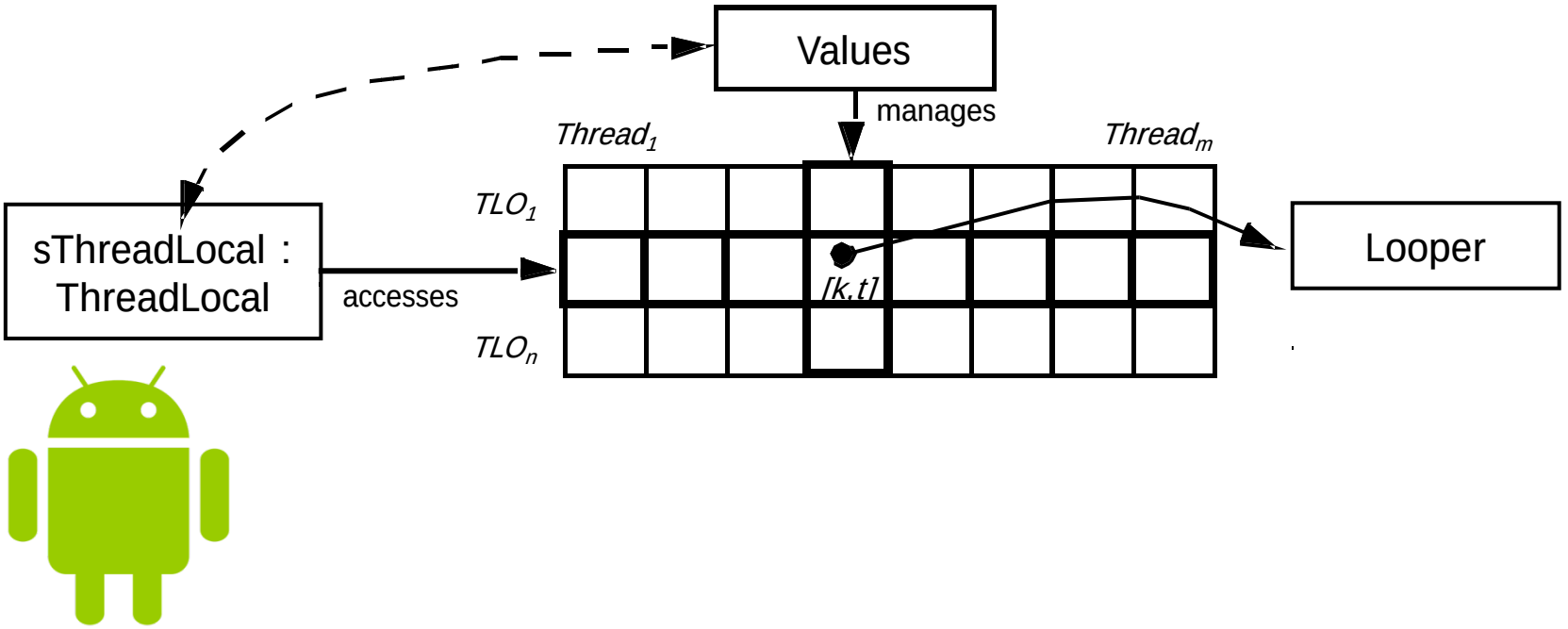




www.dre.vanderbilt.edu/~schmidt/PDF/TSS-pattern.pdf has more info

Patterns Applied in Android Frameworks

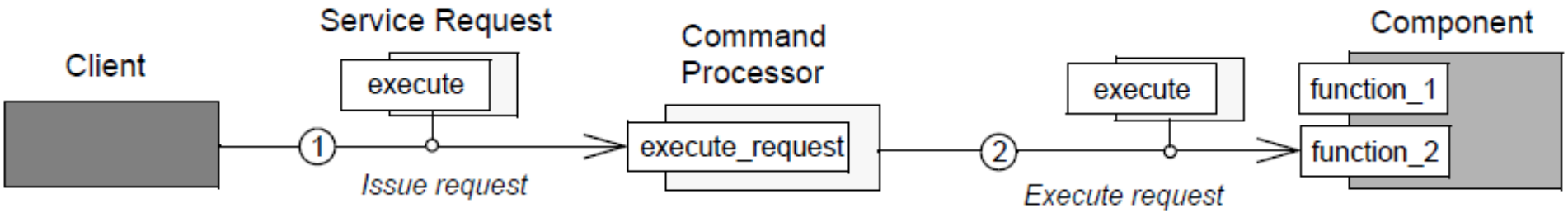
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Thread-Specific Storage* – allows multiple threads to use one 'logically global' access point to retrieve an object that is local to a thread, without incurring locking overhead on each object access

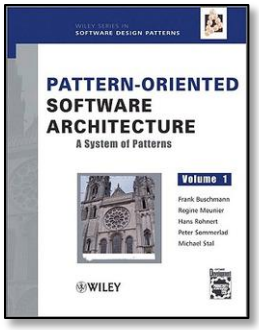


See upcoming parts on "The *Thread-Specific Storage* Pattern"

Patterns In Android Concurrency Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Command Processor* – package a piece of application functionality—as well as its parameterization in an object—to execute it in another context
    - e.g., at a later point in time, in a different process or thread, etc.

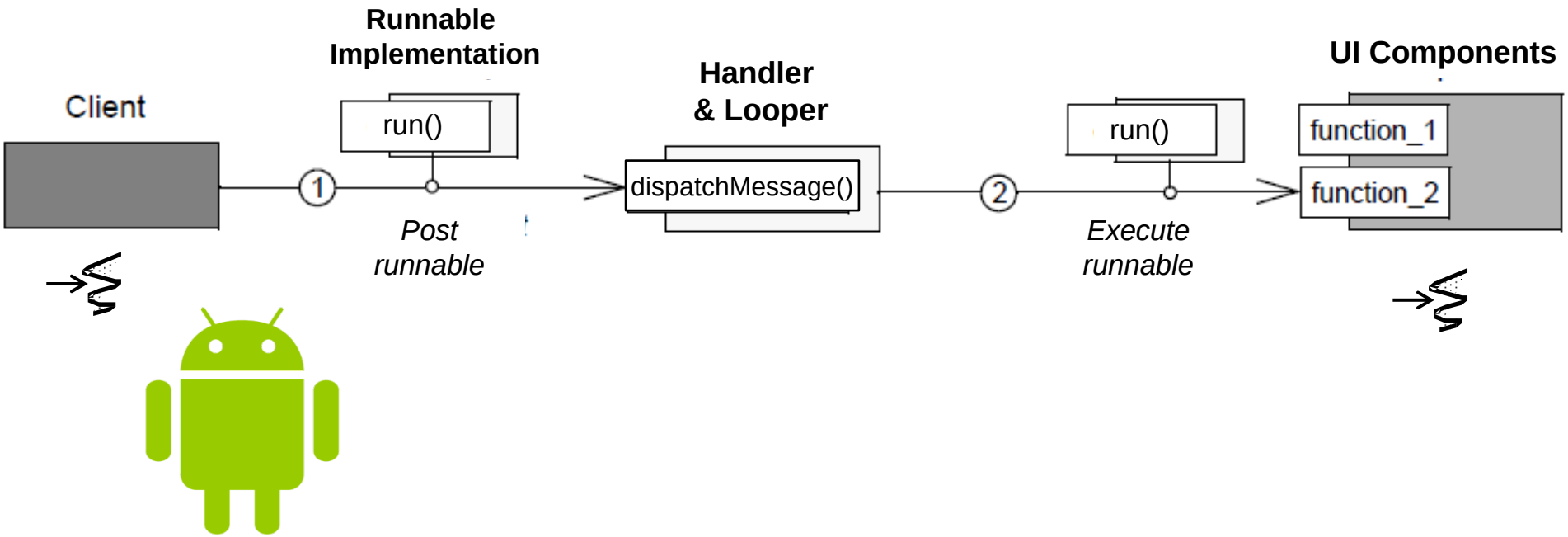




www.dre.vanderbilt.edu/~schmidt/PDF/CommandRevisited.pdf has more info

Patterns In Android Concurrency Frameworks

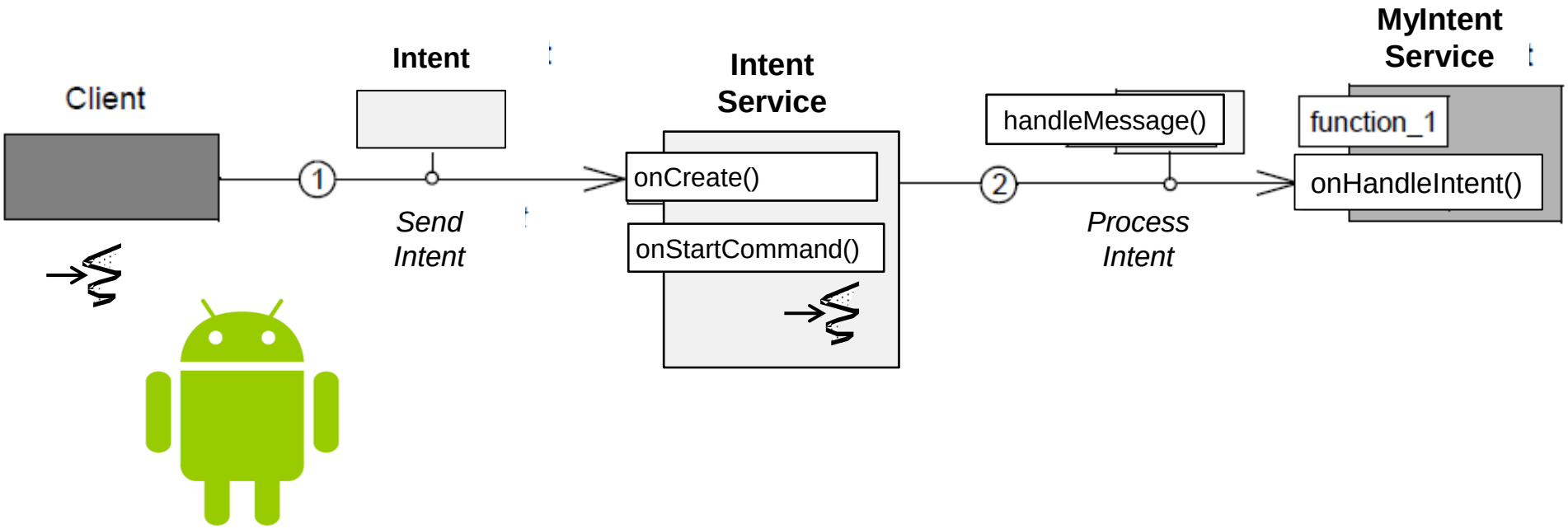
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Command Processor* – package a piece of application functionality—as well as its parameterization in an object—to execute it in another context
    - e.g., at a later point in time, in a different process or thread, etc.



See upcoming parts on the "Android Handler"

Patterns In Android Concurrency Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Command Processor* – package a piece of application functionality—as well as its parameterization in an object—to execute it in another context
    - e.g., at a later point in time, in a different process or thread, etc.

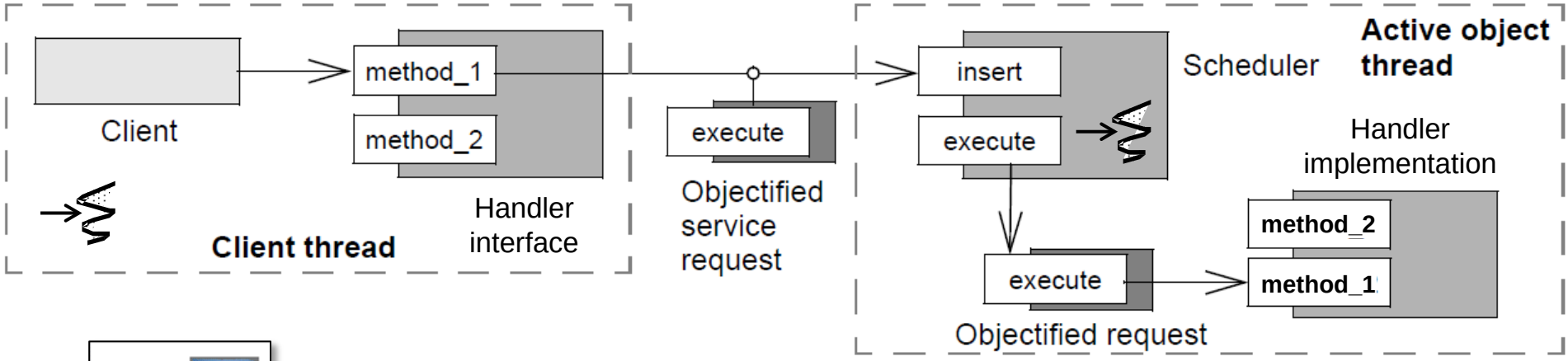


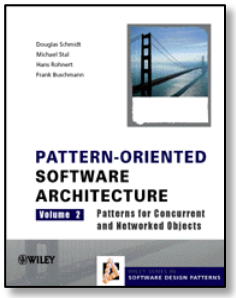
See upcoming part on the "Activity & Service Communication"

POSA Patterns in Android Concurrency Frameworks

Patterns In Android Concurrency Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Active Object* – define service requests on components as the units of concurrency & run service requests on a component in different thread(s) from the requesting client thread

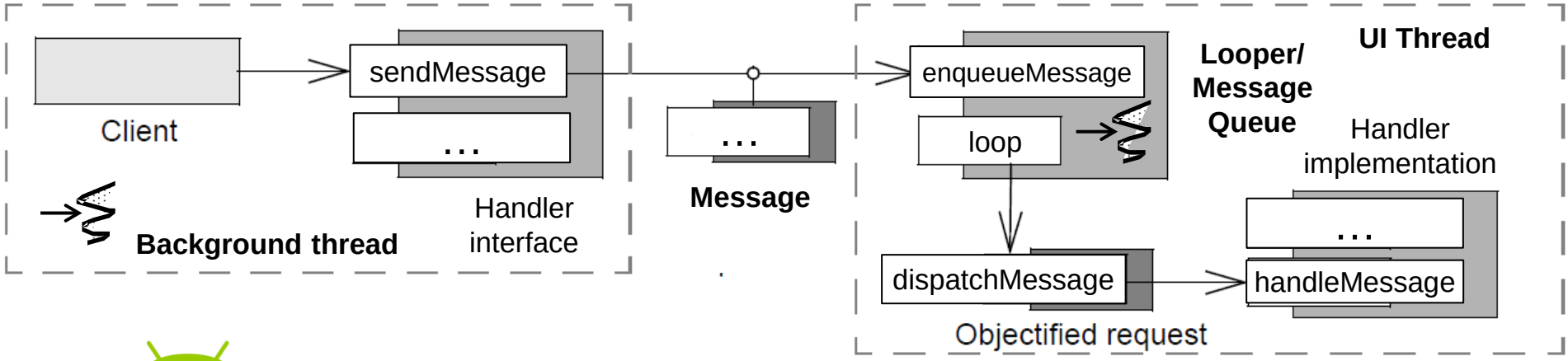




www.dre.vanderbilt.edu/~schmidt/PDF/Act-Obj.pdf has more info

Patterns In Android Concurrency Frameworks

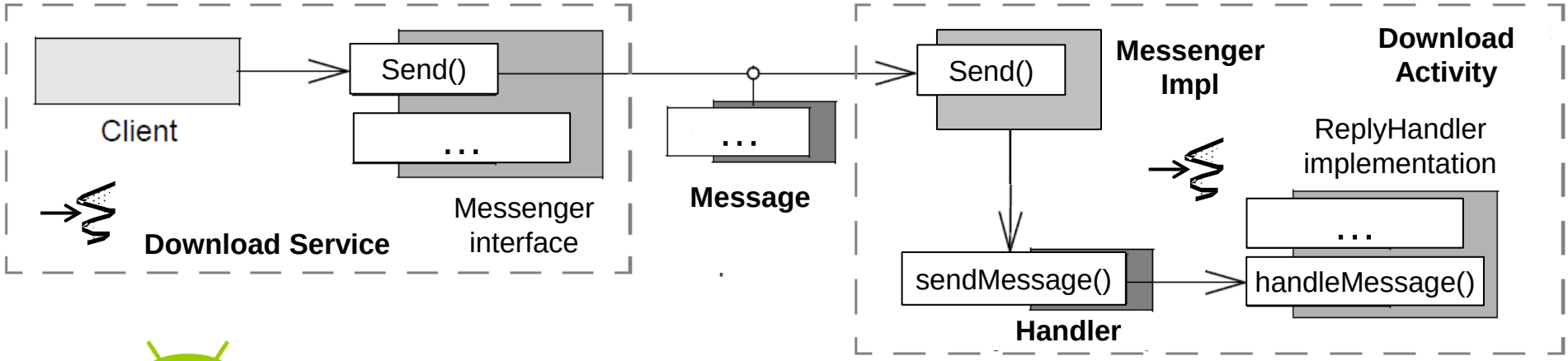
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Active Object* – define service requests on components as the units of concurrency & run service requests on a component in different thread(s) from the requesting client thread



See upcoming parts on "The *Active Object* Pattern"

Patterns In Android Concurrency Frameworks

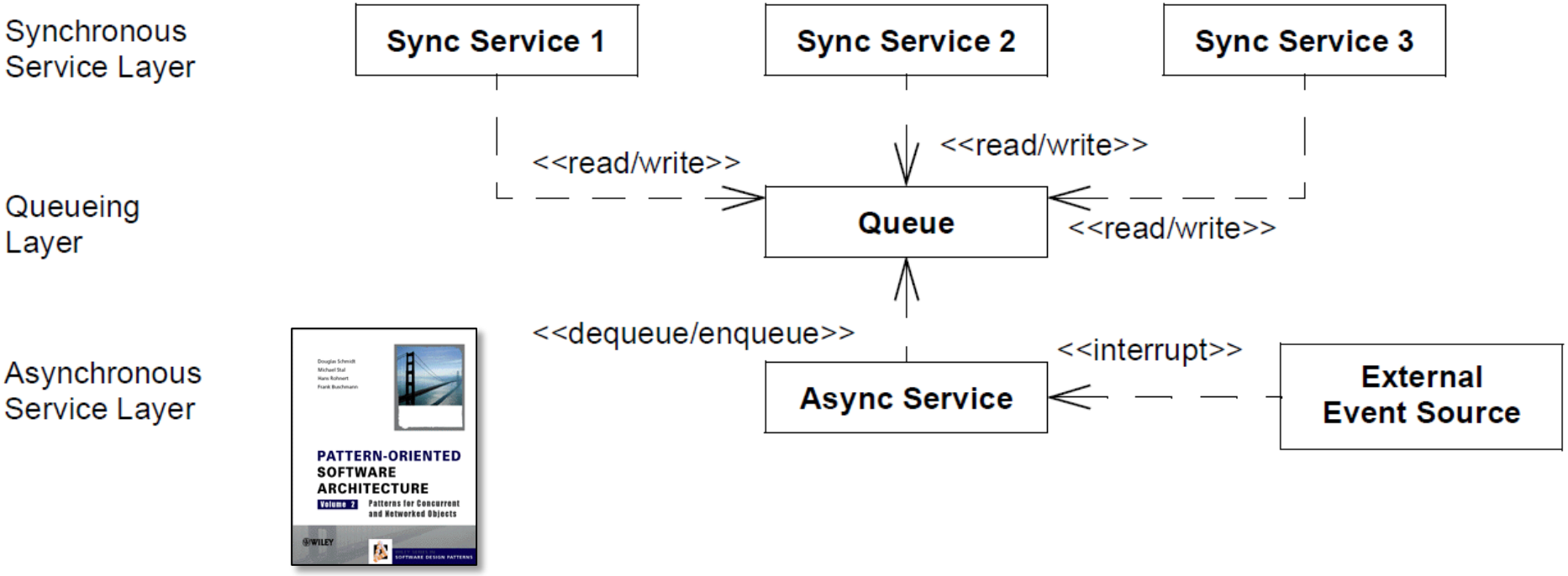
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Active Object* – define service requests on components as the units of concurrency & run service requests on a component in different thread(s) from the requesting client thread



See upcoming part on "Activity & Service Communication"

Patterns In Android Concurrency Frameworks

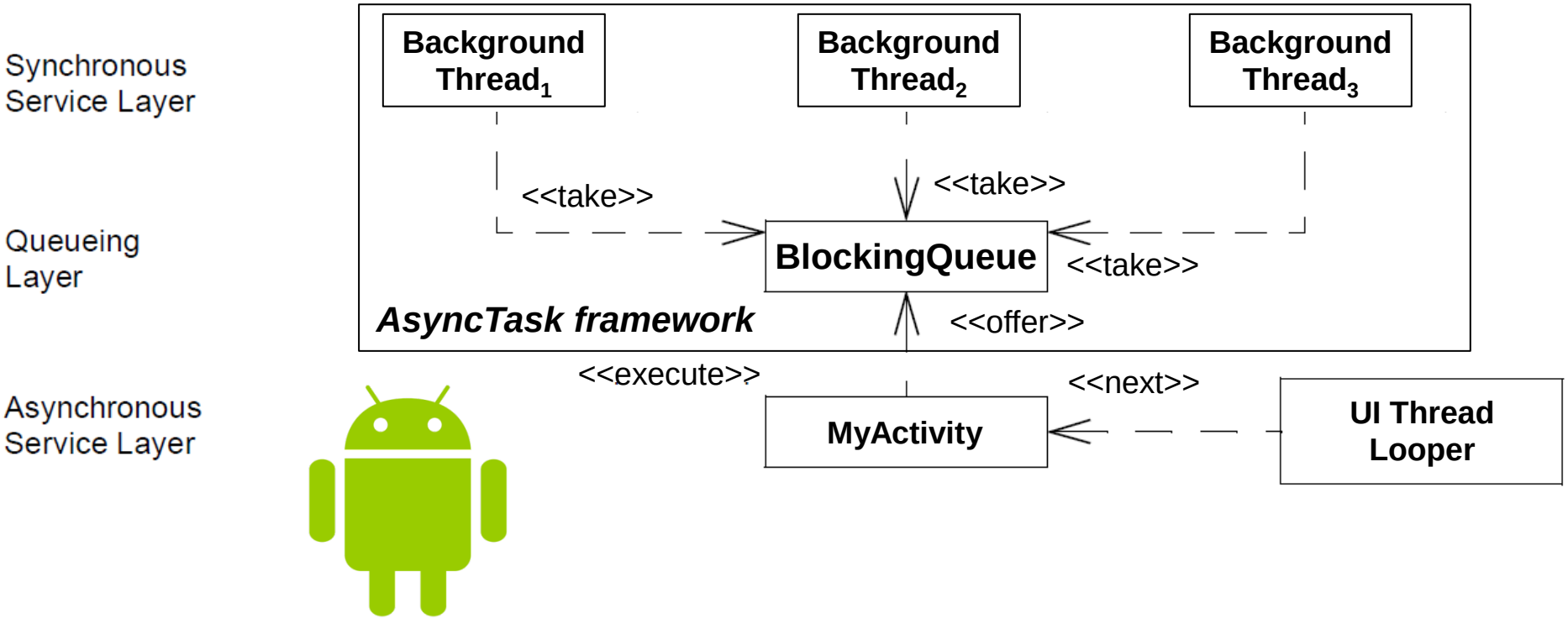
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Half-Sync/Half-Async* – Decouple async & sync service processing in concurrent systems by introducing two intercommunicating layers to simplify programming without unduly reducing performance



www.dre.vanderbilt.edu/~schmidt/PDF/HS-HA.pdf has more info

Patterns In Android Concurrency Frameworks

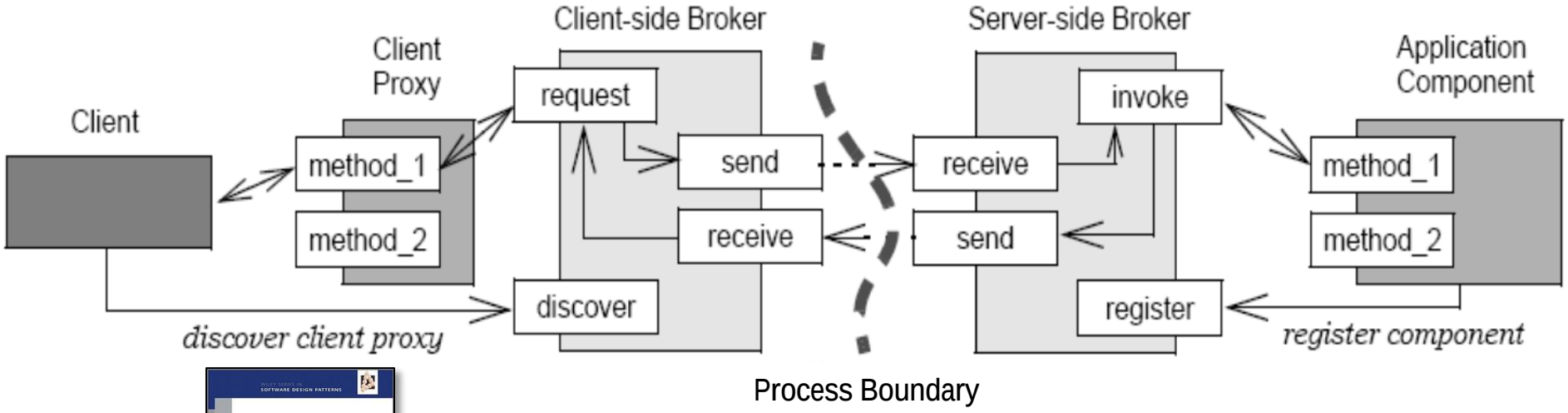
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Half-Sync/Half-Async* – Decouple async & sync service processing in concurrent systems by introducing two intercommunicating layers to simplify programming without unduly reducing performance

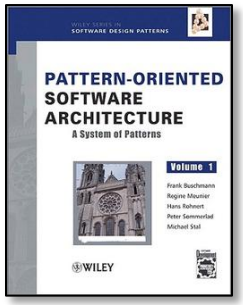


See upcoming parts on "The *Half-Sync/Half-Async* Pattern"

Patterns In Android Concurrency Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Broker* – Connect clients with remote objects by mediating invocations from clients to remote objects, while encapsulating the details of local and/or remote inter-process communication (IPC)

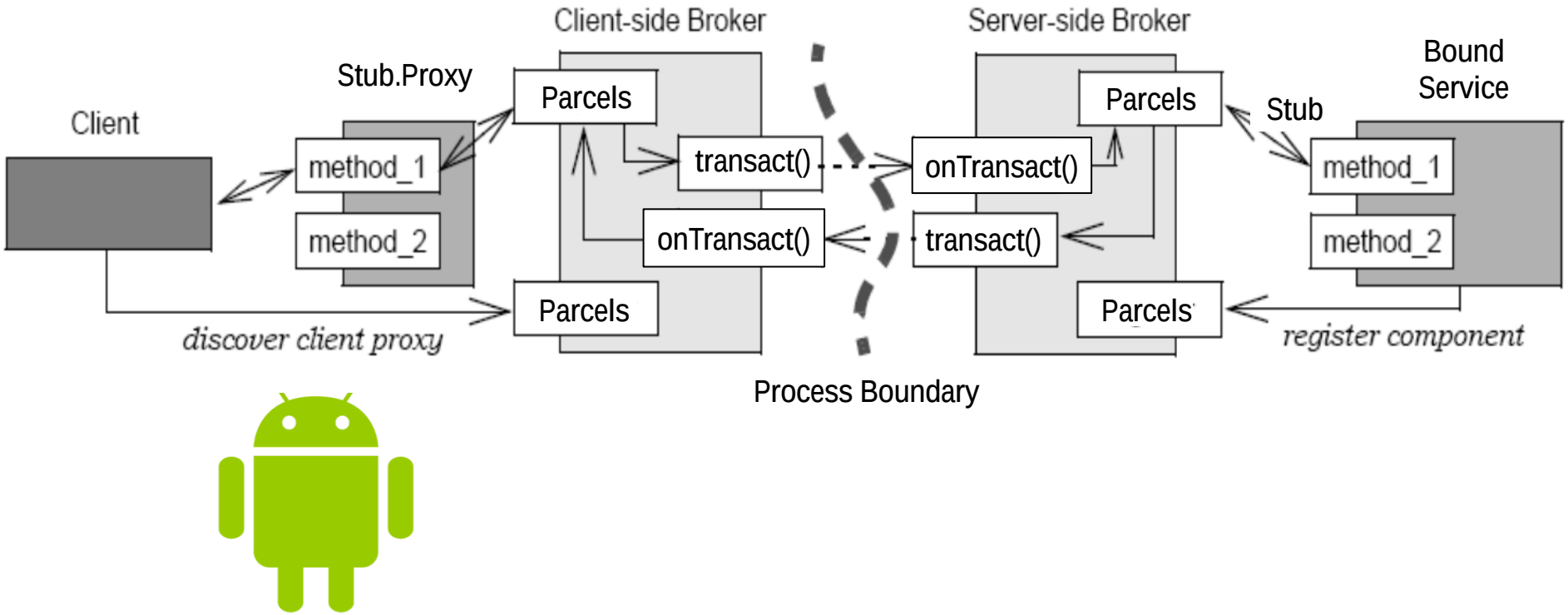




www.kircher-schwanninger.de/michael/publications/BrokerRevisited.pdf

Patterns In Android Concurrency Frameworks

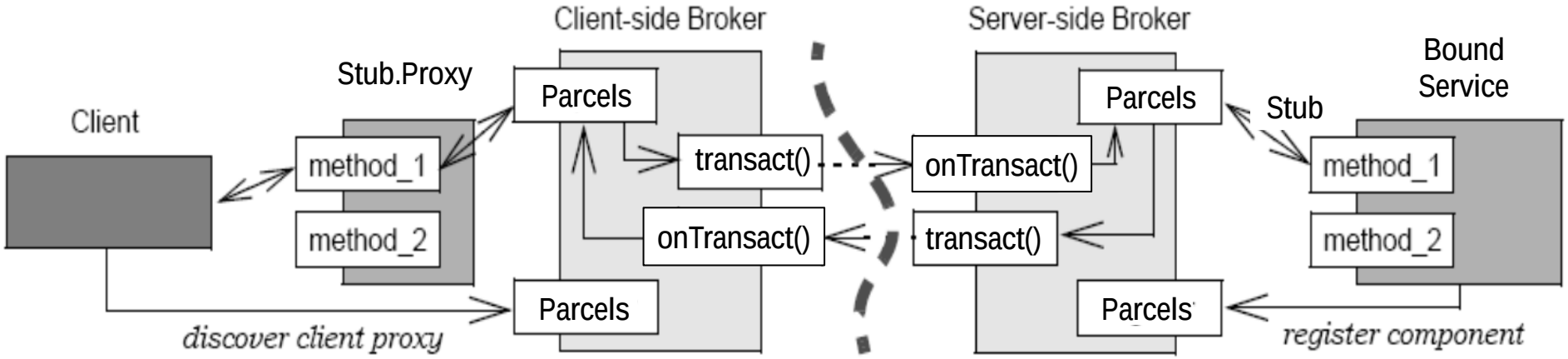
- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Broker* – Connect clients with remote objects by mediating invocations from clients to remote objects, while encapsulating the details of local and/or remote inter-process communication (IPC)



See upcoming part on "Activity & Service Communication"

Patterns In Android Concurrency Frameworks

- Android's concurrency & communication frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Broker* – Connect clients with remote objects by mediating invocations from clients to remote objects, while encapsulating the details of local and/or remote inter-process communication (IPC)

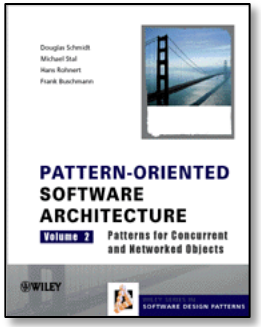


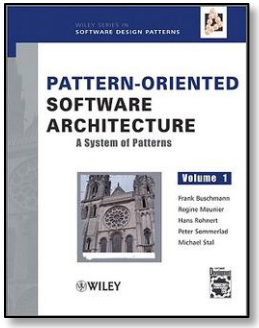
www.dre.vanderbilt.edu/~schmidt/POSA has more on *Broker* pattern language

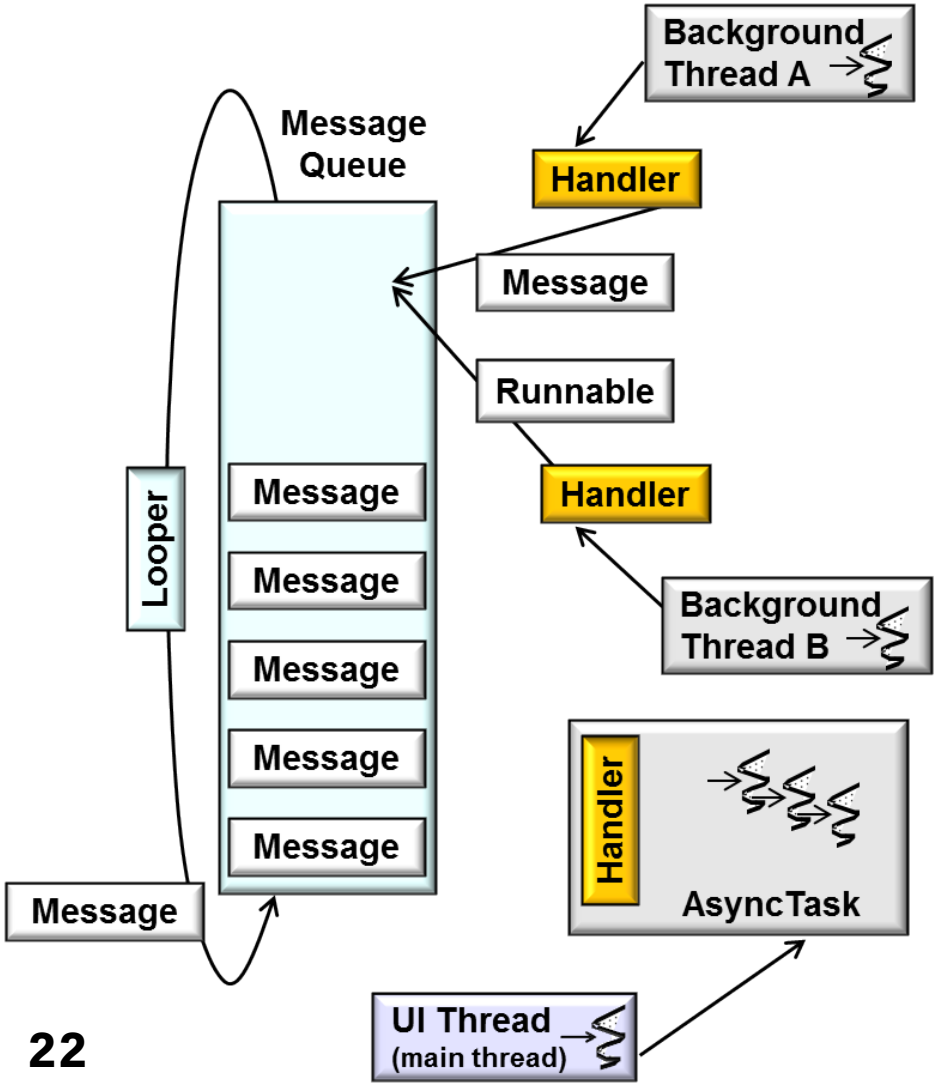
Gang-of-Four Patterns in Android Concurrency Frameworks

Patterns In Android Concurrency Frameworks

- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns

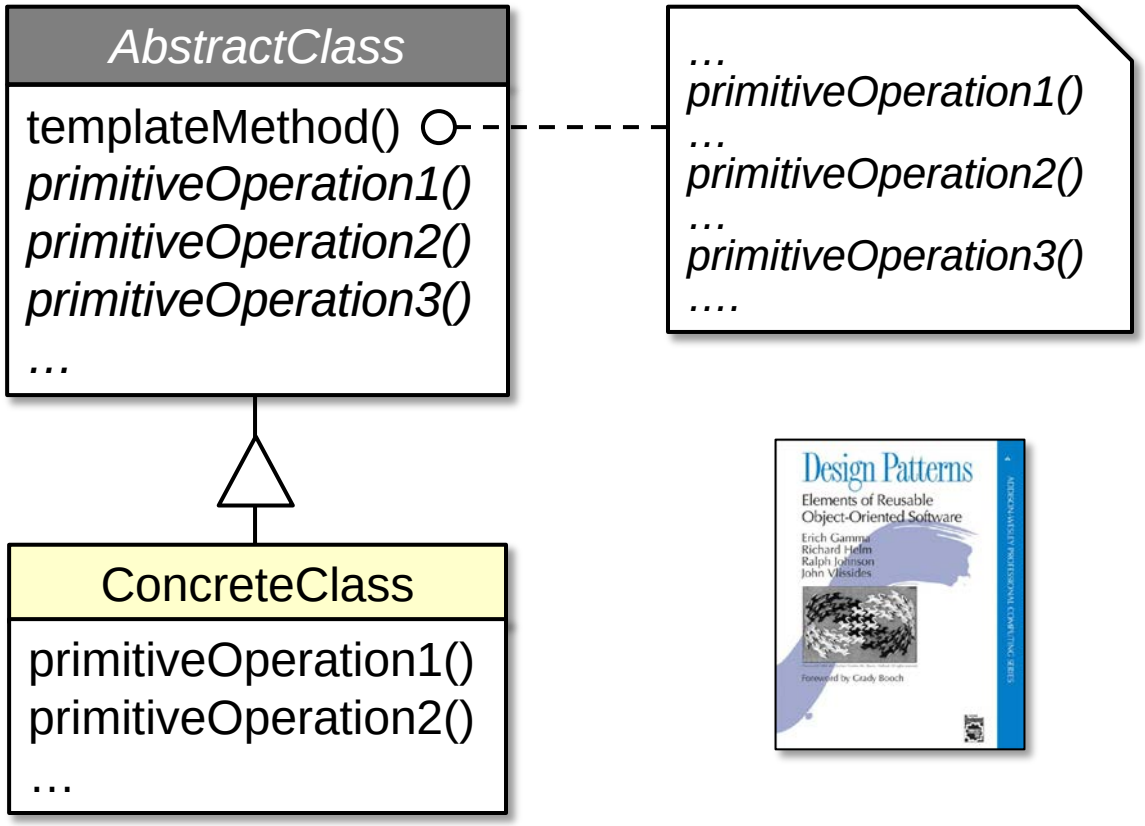






Patterns In Android Concurrency Frameworks

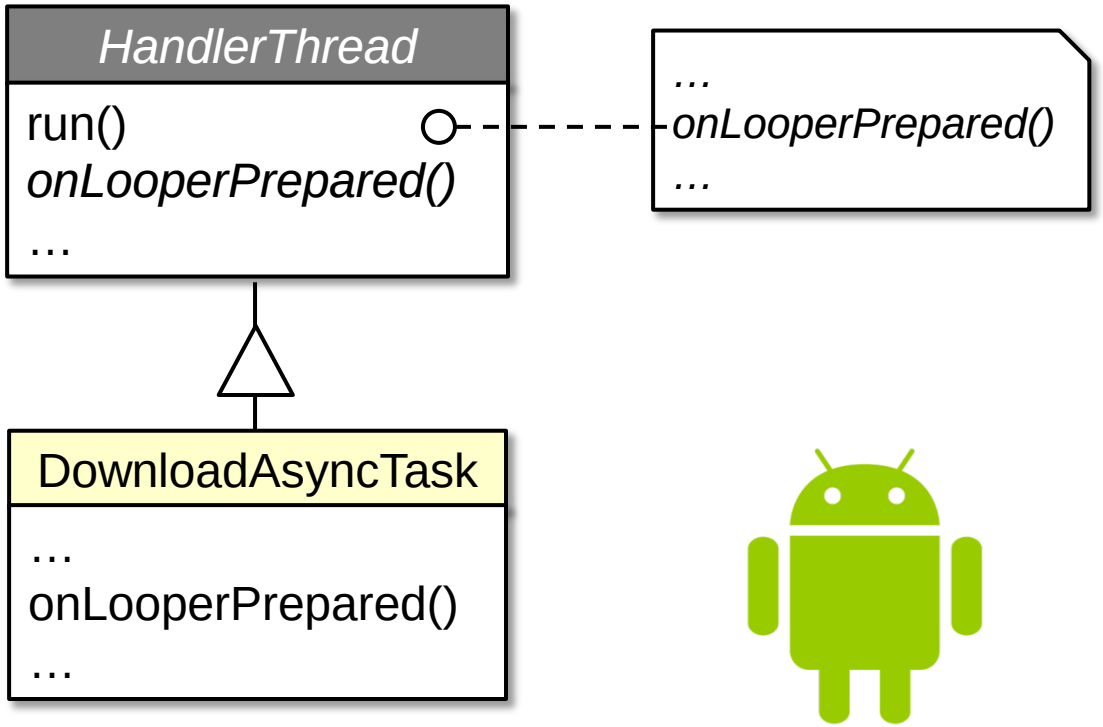
- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Template Method* – provide a skeleton of an algorithm in a method, deferring some steps to subclasses



en.wikipedia.org/wiki/Template_method_pattern has more info

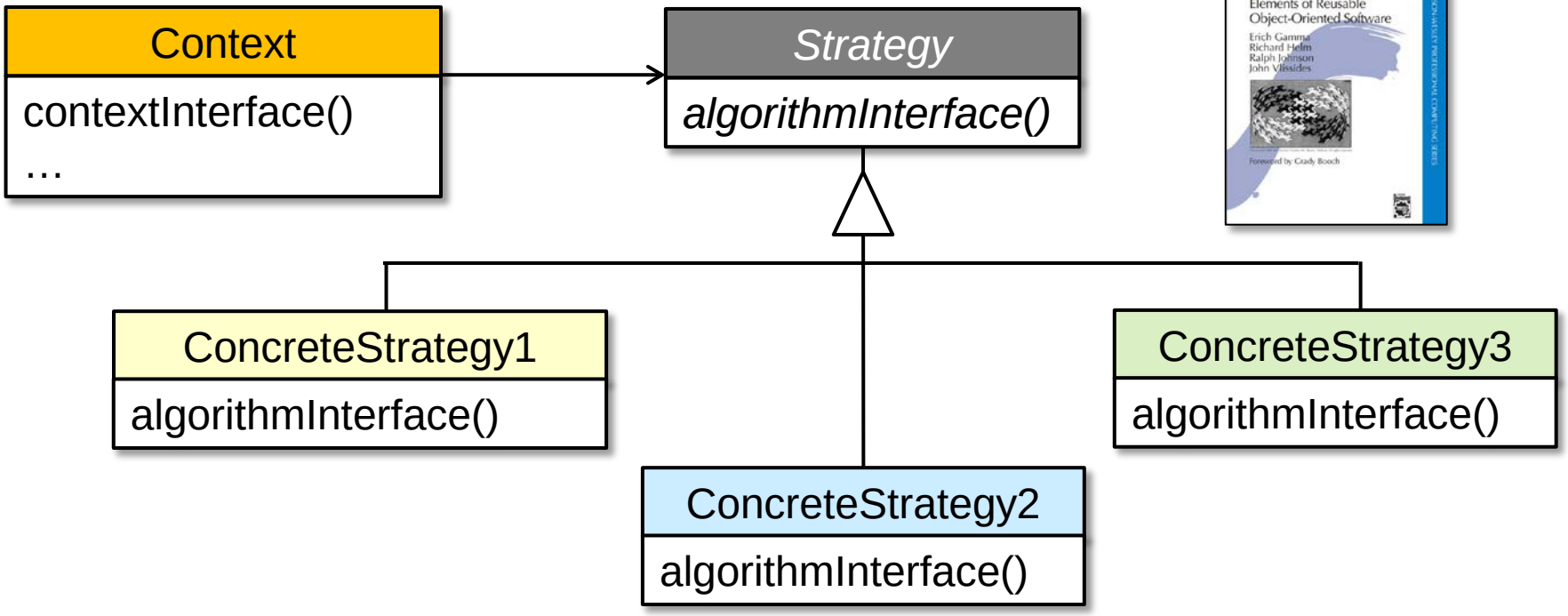
Patterns In Android Concurrency Frameworks

- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Template Method* – provide a skeleton of an algorithm in a method, deferring some steps to subclasses



Patterns In Android Concurrency Frameworks

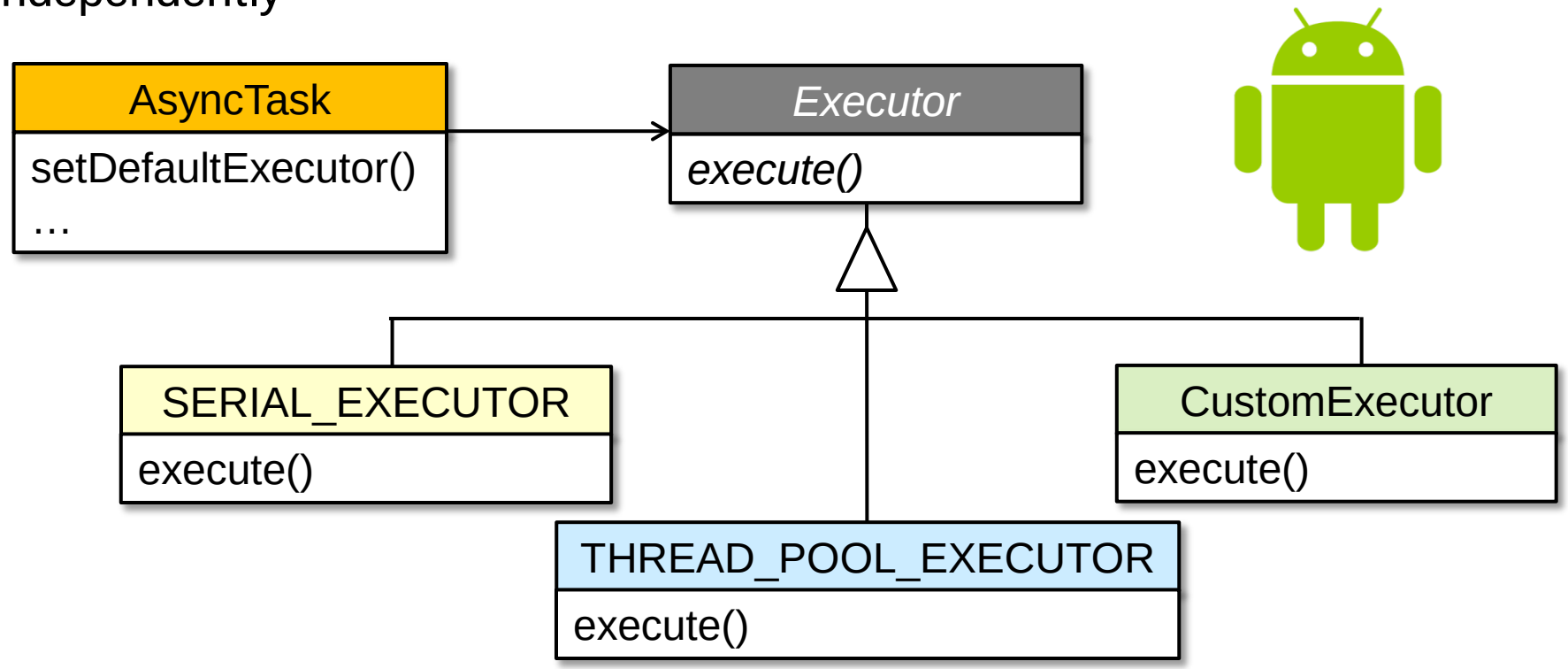
- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Strategy* – define a family of algorithms, encapsulate each one, & make them interchangeable to let clients & algorithms vary independently



en.wikipedia.org/wiki/Strategy_pattern has more info

Patterns In Android Concurrency Frameworks

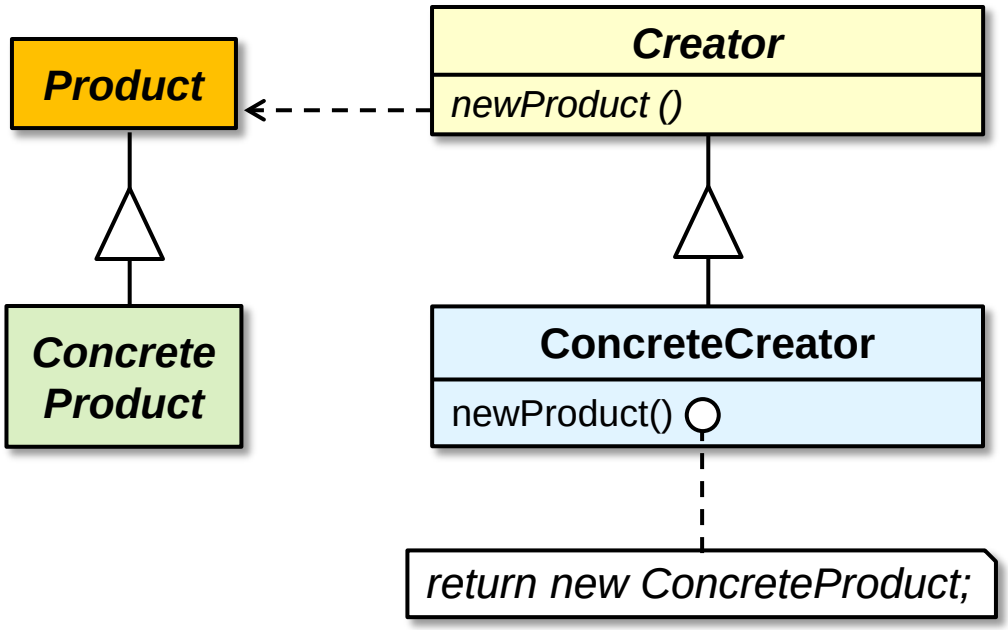
- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Strategy* – define a family of algorithms, encapsulate each one, & make them interchangeable to let clients & algorithms vary independently



See upcoming part on the "AsyncTask Framework"

Patterns In Android Concurrency Frameworks

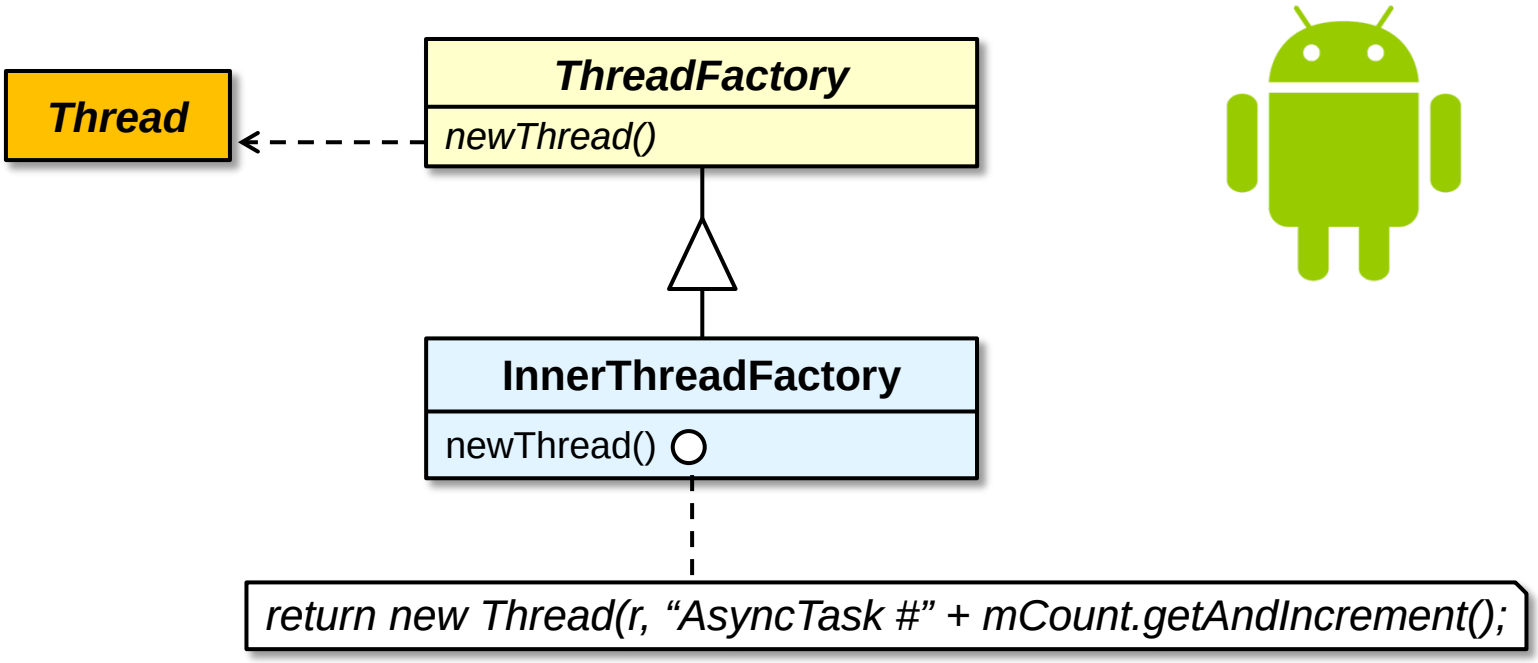
- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Factory Method* – provide an interface for creating an object, but leaving the choice of the object's concrete type to a subclass



See en.wikipedia.org/wiki/Factory_method_pattern for more info

Patterns In Android Concurrency Frameworks

- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - *Factory Method* – provide an interface for creating an object, but leaving the choice of the object's concrete type to a subclass



See upcoming parts on "The *Half-Sync/Half-Async* Pattern"

Patterns In Android Concurrency Frameworks

- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - Java concurrency mechanisms are used in Android's concurrent frameworks

package **Added in API level 1**

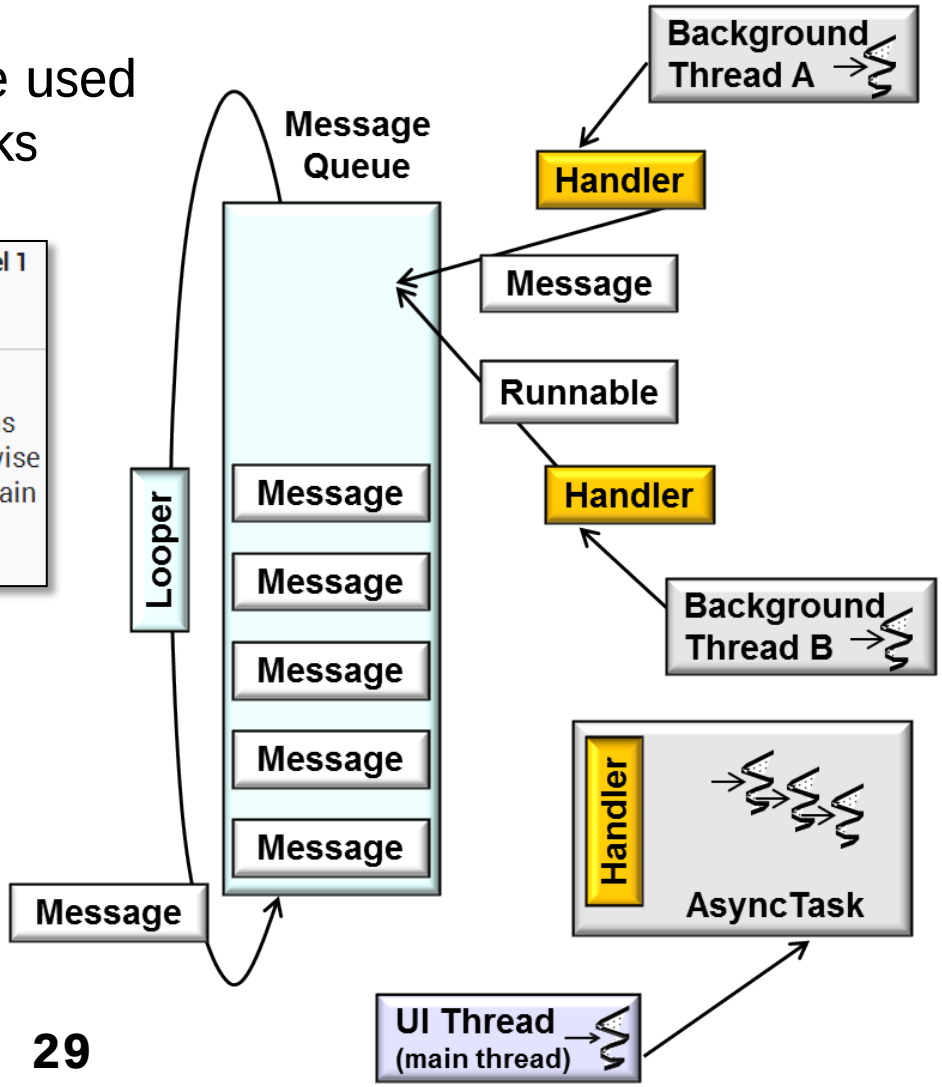
java.util.concurrent

Utility classes commonly useful in concurrent programming. This package includes a few small standardized extensible frameworks, as well as some classes that provide useful functionality and are otherwise tedious or difficult to implement. Here are brief descriptions of the main components. See also the `java.util.concurrent.locks` and `java.util.concurrent.atomic` packages.

package **Added in API level 1**

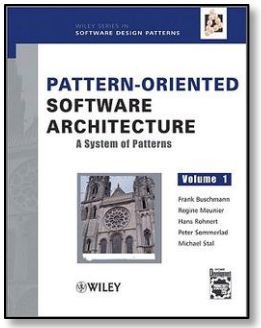
java.util.concurrent.locks

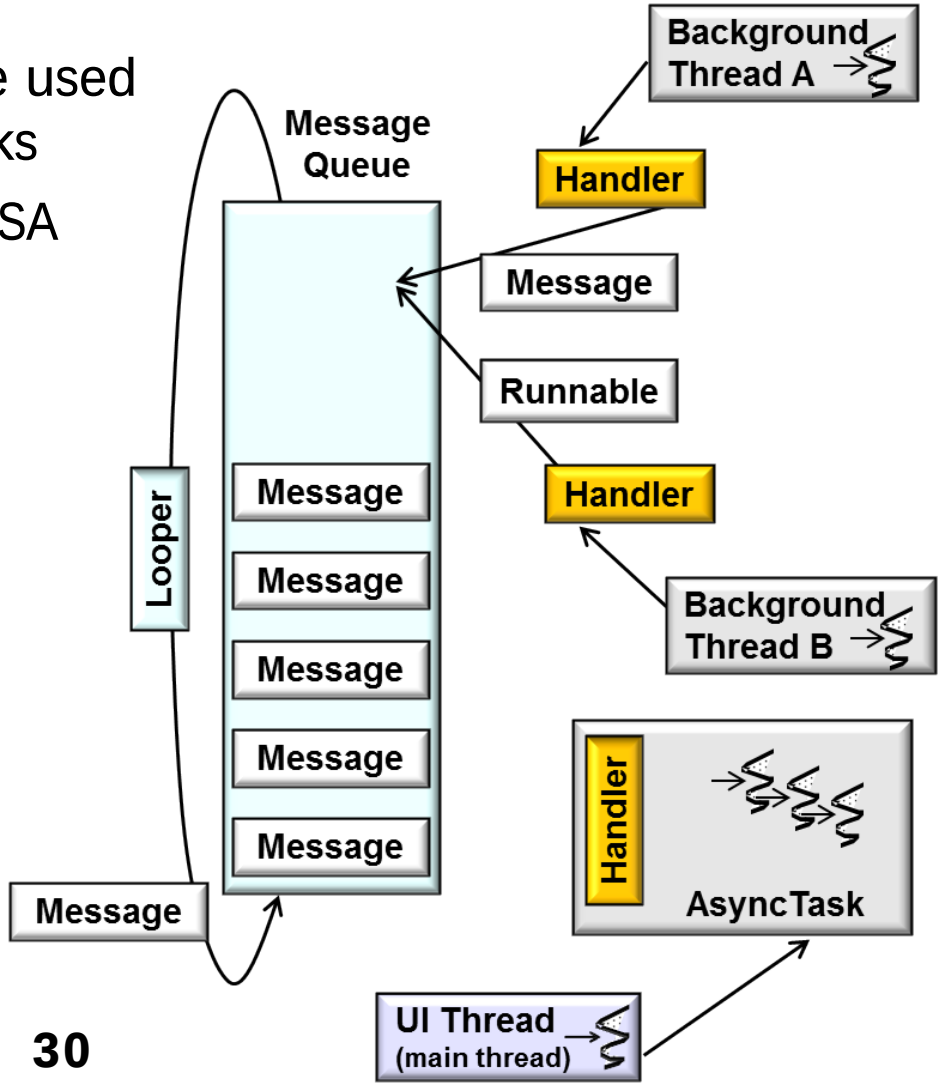
Interfaces and classes providing a framework for locking and waiting for conditions that is distinct from built-in synchronization and monitors. The framework permits much greater flexibility in the use of locks and conditions, at the expense of more awkward syntax.
The `Lock` interface supports locking disciplines that differ in semantics (reentrant, fair, etc), and that can be used in non-block-structured contexts including hand-over-hand and lock reordering algorithms. The main implementation is `ReentrantLock`.



Patterns In Android Concurrency Frameworks

- Android's concurrency frameworks are designed, implemented, & integrated in accordance with many patterns
  - Java concurrency mechanisms are used in Android's concurrent frameworks
  - We'll also discuss other GoF & POSA patterns throughout this section

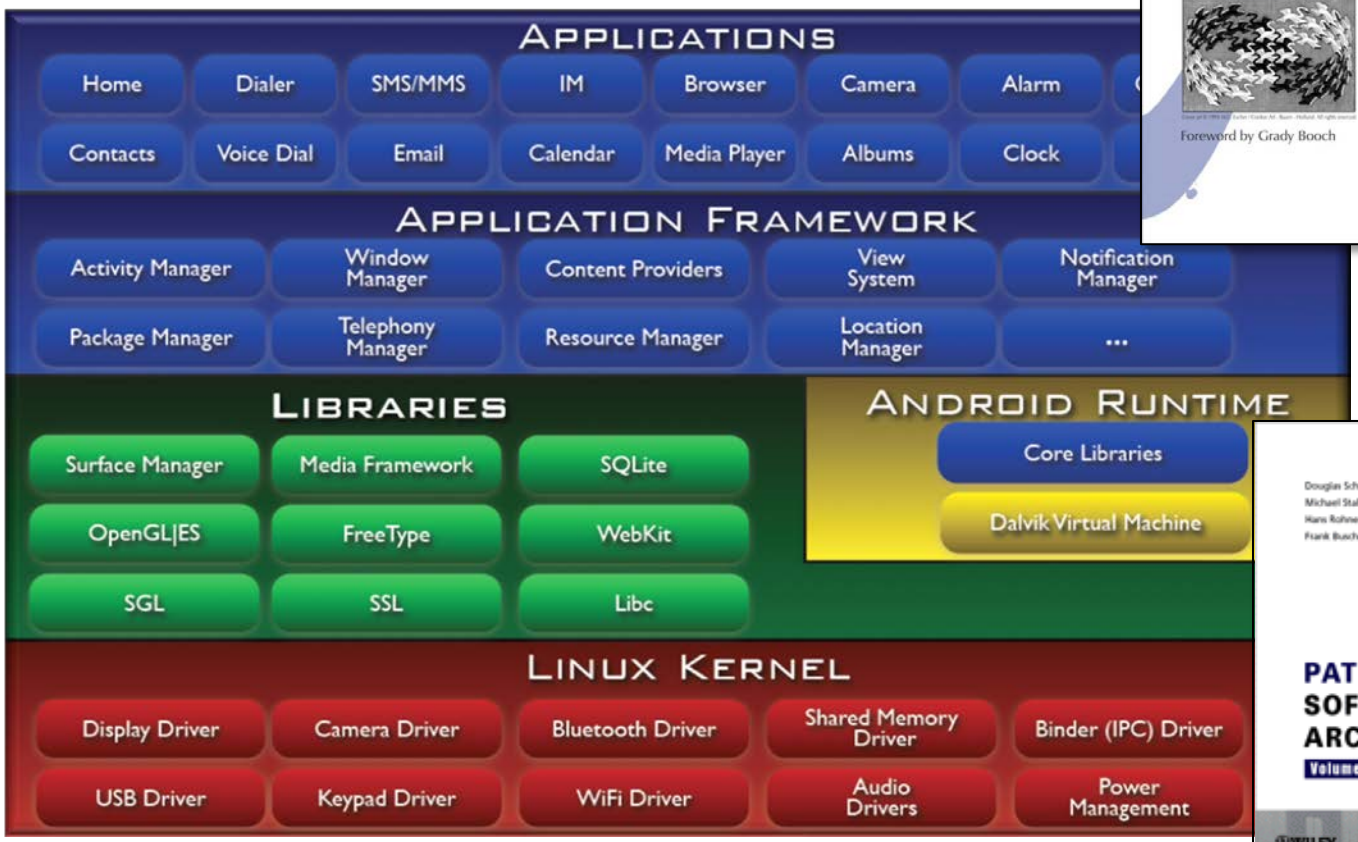


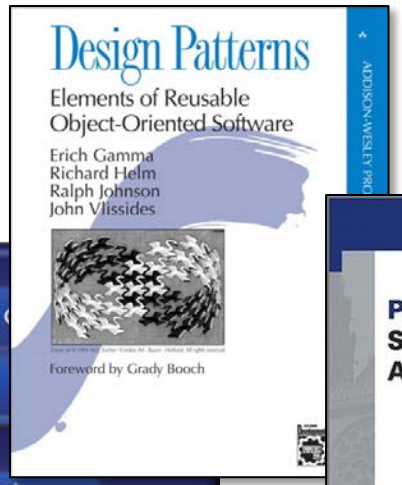


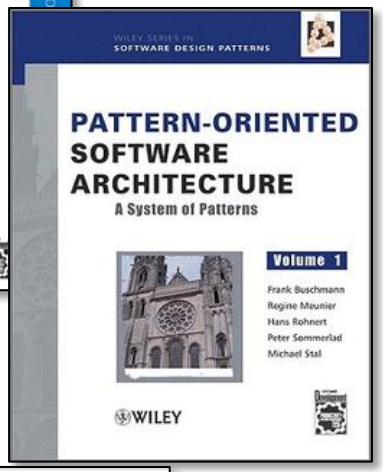
Summary

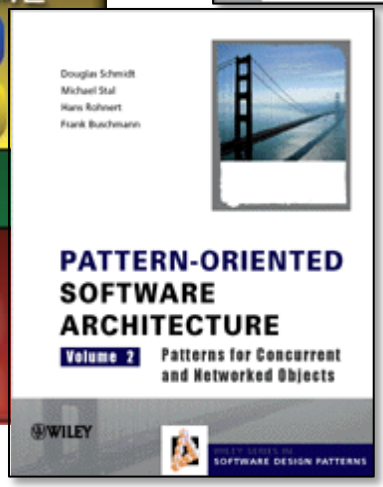
Summary

- Android applies *patterns* & *frameworks* to help improve its structure & functionality



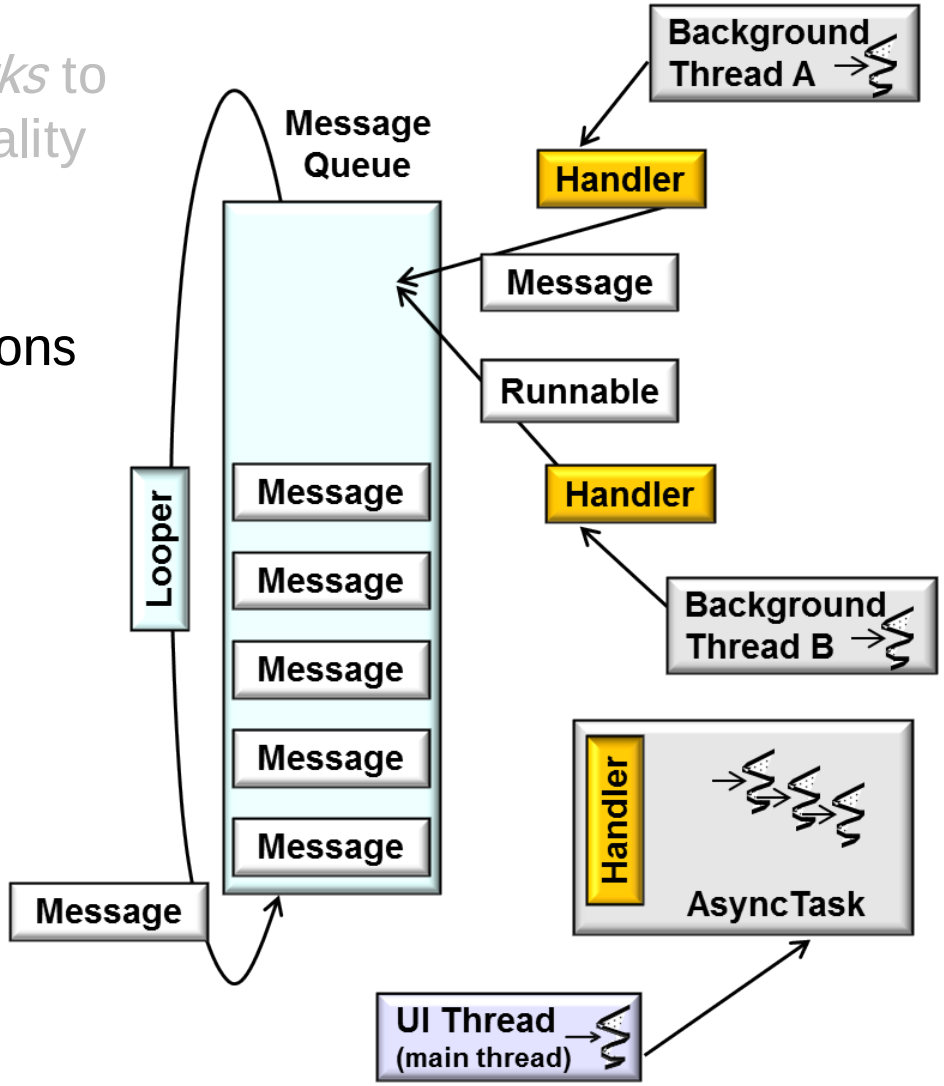






Summary

- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations



Summary

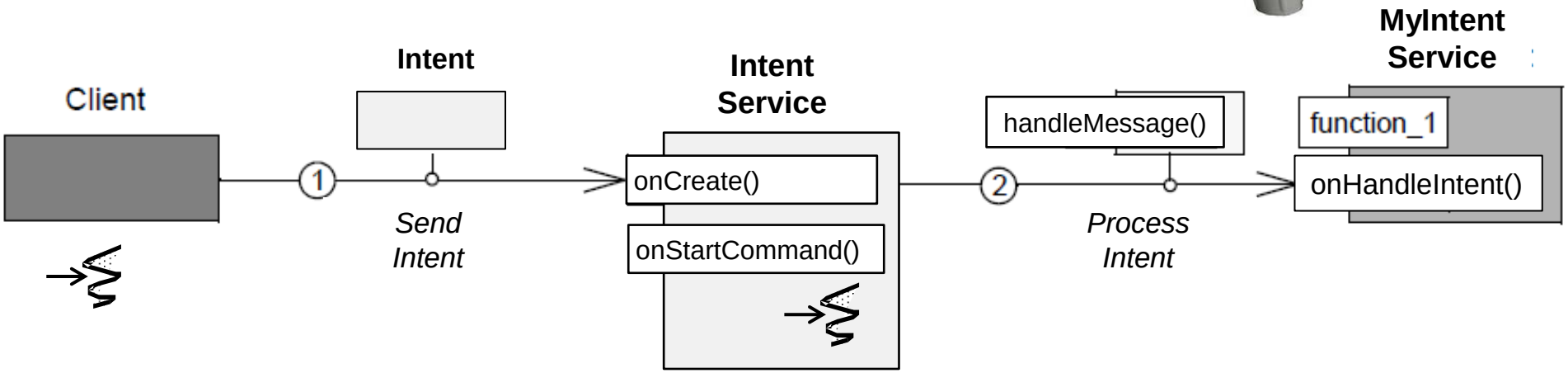
- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations

Summary

- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations

Summary

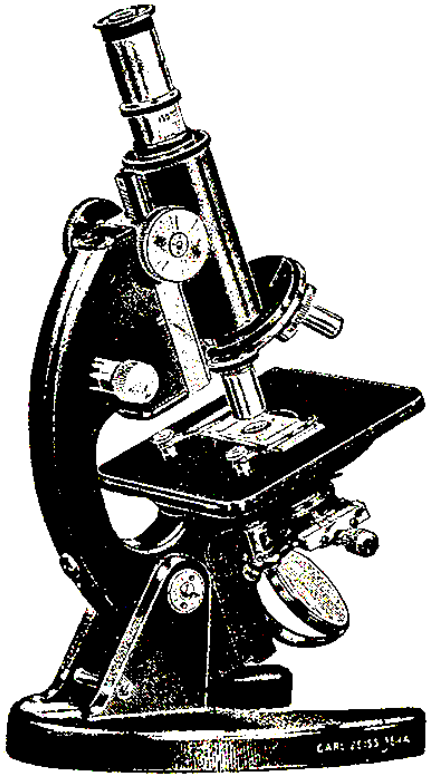
- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations
- Help developers focus on "business logic" rather than tedious & error-prone details



Summary

- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations
- Help developers focus on "business logic" rather than tedious & error-prone details
- Android's pattern-oriented, framework-based architecture greatly simplifies its software quality attributes

Summary



- Android applies *patterns* & *frameworks* to help improve its structure & functionality
- Patterns & frameworks codify "best practices" of design & architecture in systematically reusable implementations
- Help developers focus on "business logic" rather than tedious & error-prone details
- Android's pattern-oriented, framework-based architecture greatly simplifies its software quality attributes
- Since it's available as open-source it's an ideal environment for learning about—& experimenting with—powerful forms of systematic reuse

See class.coursera.org/posa-002/wiki/Source_Code_By_Week