

Overview of Reactive Programming Principles

Douglas C. Schmidt

d.schmidt@vanderbilt.edu

www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

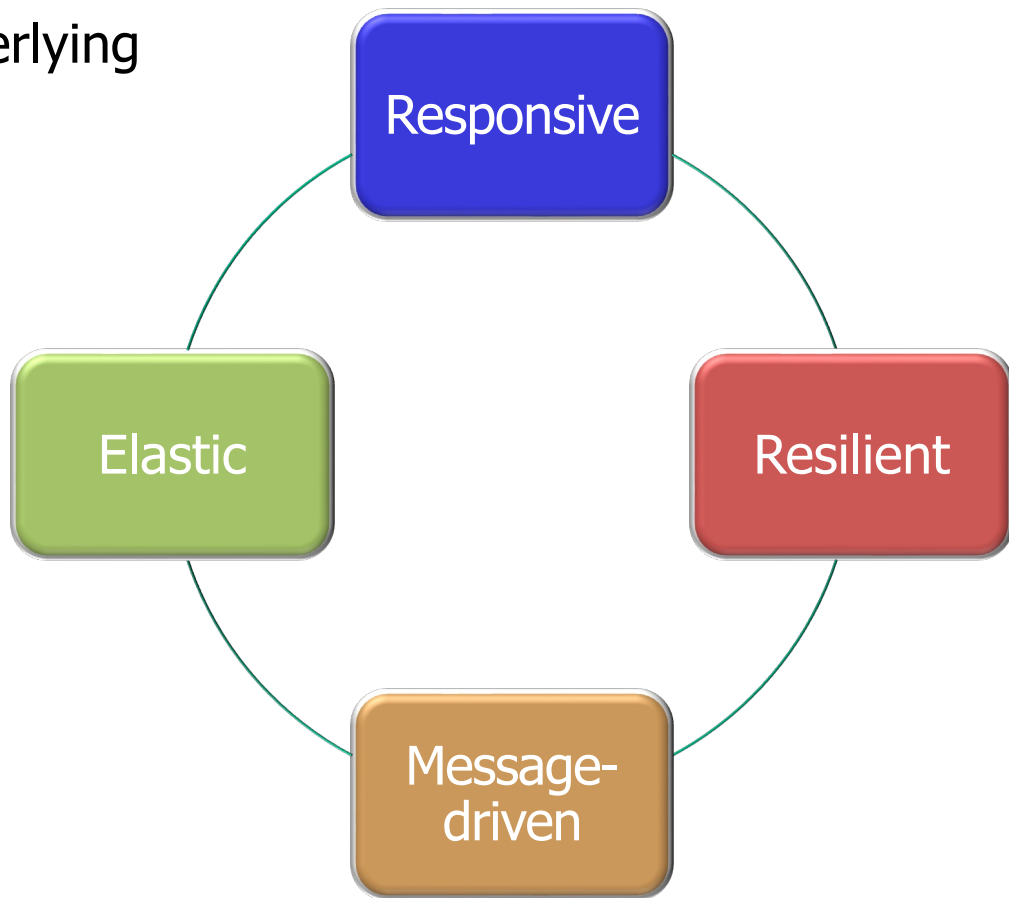
**Institute for Software
Integrated Systems**

**Vanderbilt University
Nashville, Tennessee, USA**



Learning Objectives in this Part of the Lesson

- Understand the key principles underlying reactive programming



Overview of Reactive Programming

Overview of Reactive Programming

- Reactive programming is an asynchronous programming paradigm concerned with processing data streams & propagation of changes



See en.wikipedia.org/wiki/Reactive_programming

Overview of Reactive Programming

- Reactive programming is particularly useful to support certain scenarios



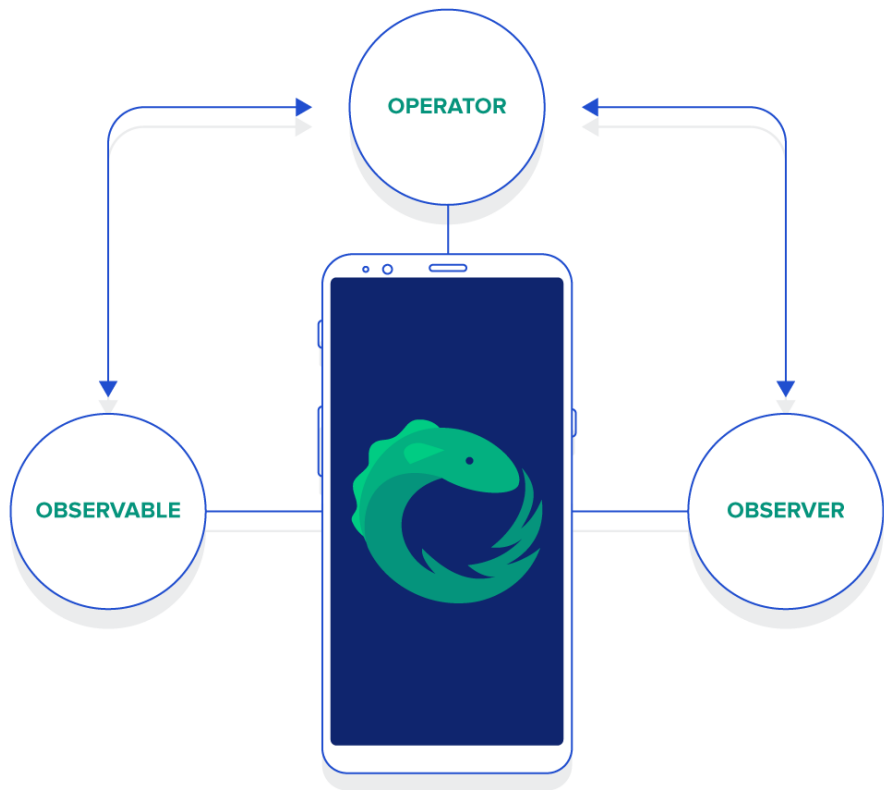
Overview of Reactive Programming

- Reactive programming is particularly useful to support certain scenarios, e.g.
 - Processing user events



Overview of Reactive Programming

- Reactive programming is particularly useful to support certain scenarios, e.g.
 - Processing user events
 - e.g., mouse movement/clicks, touch events, GPS location signals, etc.



See github.com/ReactiveX/RxAndroid

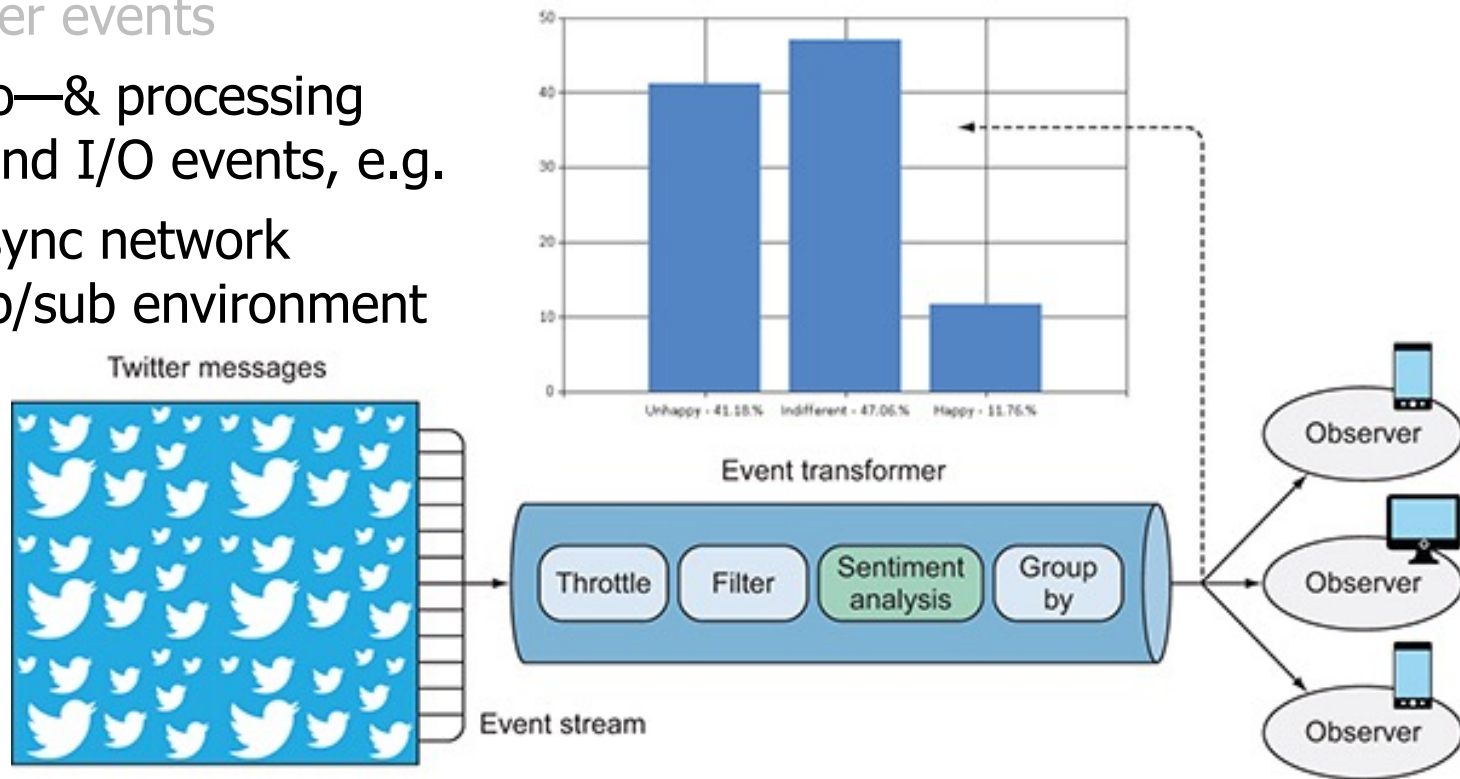
Overview of Reactive Programming

- Reactive programming is particularly useful to support certain scenarios, e.g.
 - Processing user events
 - Responding to—and processing—latency-bound I/O events



Overview of Reactive Programming

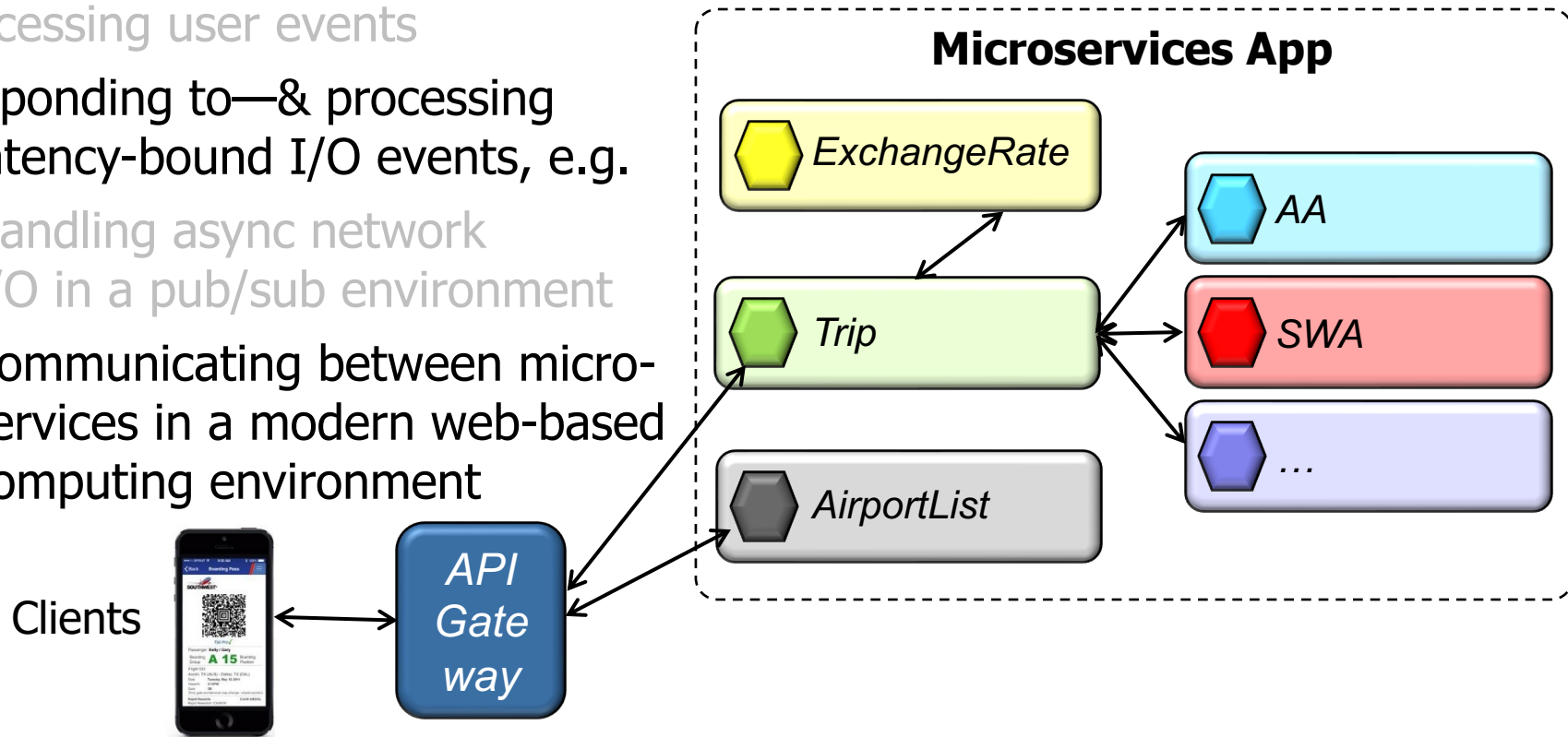
- Reactive programming is particularly useful to support certain scenarios, e.g.
 - Processing user events
 - Responding to—and processing—latency-bound I/O events, e.g.
 - Handling async network I/O in a pub/sub environment



See www.youtube.com/watch?v=z0a0N9OgaAA

Overview of Reactive Programming

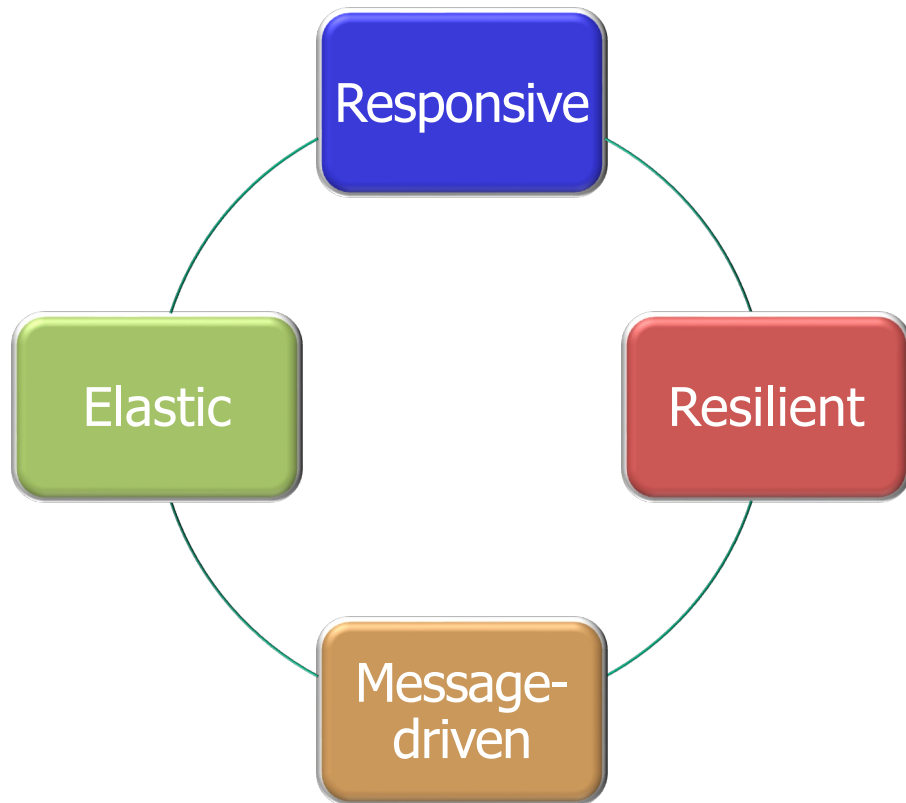
- Reactive programming is particularly useful to support certain scenarios, e.g.
 - Processing user events
 - Responding to—and processing—latency-bound I/O events, e.g.
 - Handling async network I/O in a pub/sub environment
 - Communicating between microservices in a modern web-based computing environment



See docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html

Overview of Reactive Programming

- Reactive programming is based on four key principles



See www.reactivemanifesto.org

Overview of Reactive Programming

- Reactive programming is based on four key principles, e.g.

- Responsive**

- Provide rapid & consistent response times



Establish reliable upper bounds to deliver consistent quality of service & prevent delays

See en.wikipedia.org/wiki/Responsiveness

Overview of Reactive Programming

- Reactive programming is based on four key principles, e.g.

- Responsive**

- Resilient**

- The system remains responsive, even in the face of failure



Failure of some operations should not bring the entire system down

See [en.wikipedia.org/wiki/Resilience_\(network\)](https://en.wikipedia.org/wiki/Resilience_(network))

Overview of Reactive Programming

- Reactive programming is based on four key principles, e.g.

- Responsive**

- Resilient**

- Elastic**

- A system should remain responsive, even under varying workload

It should be possible to "auto-scale" performance



See en.wikipedia.org/wiki/Autoscaling

Overview of Reactive Programming

- Reactive programming is based on four key principles, e.g.

- Responsive**

This principle is an "implementation detail" wrt the others..

- Resilient**

- Elastic**

- Message-driven**

- Asynchronous message-passing ensures loose coupling, isolation, & location transparency between components



See en.wikipedia.org/wiki/Message-oriented_middleware

End of Overview of Reactive Programming Principles