PL@NES - Reading Club

Paper: Deprecating the Observer Pattern with Scala.React
by Ingo Maier & Martin Odersky

Christophe.VanGinneken@cs.kuleuven.be
THE FOLLOWING **PRESENTATION** DOES NOT INCLUDE MATERIAL ABOUT **SCALA** NOR **SCALA.REACT**. IT WILL HIGHLIGHT THE GREAT PARTS OF THE ORIGINAL PAPER IN A GENERIC WAY.
Contributions

- Observer Pattern
- Reactive Semantics
- Scala React (ok, maybe a little ;-)
- Reactive Performance

Goals
- summary of RP
- applicability to NES
Observer Pattern

- Subject
  - add()
  - remove()
  - notify()

- Observer
  - update()

- ConcreteSubject

- ConcreteObserver
  - update()

update() obtains state information from the subject and acts on that state

DEPRECATED
Observer Pattern

```kotlin
var path: Path = null

val moveObserver = { (event: MouseEvent) ->
    path.lineTo(event.position)
    draw(path)
}

control.addMouseDownObserver { event ->
    path = new Path(event.position)
    control.addMouseMoveObserver(moveObserver)
}

control.addMouseUpObserver { event ->
    control.removeMouseMoveObserver(moveObserver)
    path.close()
    draw(path)
}
```

composability

encapsulation

abstraction

uniformity

semantic distance

resource management

side-effects

data consistency

separation of concerns
Reactive Semantics

- Event Streams
- Events and Signals
- Composition/Transformation
- Reactors
- Data Flow Language

observable observers
(with simple interfaces)

dependency graph

merge, map, filter, ...

Resource Mgmt
Inversion of Control

behaviour

merge, map, filter, ...

scripting of dynamic & structural
dependency graph modifications

pause, await, par, join, loop, ...

lifting

events
Reactive Semantics

+ API for dynamic binding modifications
Reactive Performance

chain

source \(\infty \rightarrow \infty \ldots \infty \rightarrow \infty\) observer

fan

source \(\infty \rightarrow \infty \rightarrow \infty\) observers

\[
\begin{array}{c|c|c|c|c|c|c|c}
\text{source} & \text{observers} & \text{relative speed} \\
\hline
\text{source} & \infty & 1.00 \quad 1.00 \\
\text{observers} & \infty & 0.71 \quad 0.74 \\
\text{relative speed} & 1.00 & 0.72 \quad 0.55 \\
\end{array}
\]
Does it Scale? on embedded devices

**Intel 2.4 GHz Intel Core 2 Duo**
2530Mhz operating frequency

**ARM 32-bit Cortex™-M3 CPU Core**
72Mhz operating frequency

Throughput (updates/second)

- **observer.c**
  - chain: 723,345
  - fan: 611,356

- **reactive-c**
  - chain: 802,978
  - fan: 518,9

Throughput (updates/second) (Spark Core)

- **observer.c**
  - chain: 9,871
  - fan: 5,189

- **reactive-c**
  - chain: 10,294
  - fan: 5,501,5

Throughput (updates/second)

- observer.c: 1,243,955 (-30%)
- reactive-c: 802,978

Throughput (updates/second) (Spark Core)

- observer.c: 9,871
- reactive-c: 5,501,5

- chain: 10,294 (-50%)
- fan: 5,189

: 110
“Time moves in one direction, memory in another.”

William Gibson
Shameless Plug

Reactive-C

```c
#include "reactive-c/api.h"
int main(void) {
    int _a = 0, _b = 0, _c = 0;
    observable_t a = observe(int, _a);
    observable_t b = observe(int, _b);
    observable_t c = observe(int, _c);
    run(
        script(
            await(a),
            await(b),
            await(delayed(all(a, b, c))),
            await(delayed(any(b, c))),
            await(all(a, b, c))
        )
    );
}
```

Rose

https://github.com/christophevg/reactive-c

https://github.com/christophevg/rose
Deprecating the Observer Pattern?

- Automate Dependency Graph Management
- Avoid Data Inconsistencies
- Offer a Functional API
- Allow for Imperative Encoding of Algorithms

Observer Pattern on Steroids!
Reactions ?
PL@NES - Reading Club

Paper: Deprecating the Observer Pattern with Scala, React
by Ingo Maier & Martin Odersky

Christophe.VanGinneken@cs.kuleuven.be

http://www.slideshare.net/christophevg/deprecating-the-observer-pattern