In the name of ALLAH

The Reactive Manifesto

A new term in Software Architecture

By

Reza Same'e < reza.samee@gmail.com >

SOFTWARE DEVELOPER @ BISPHONE

At

216th TEHLUG Session

JUNE 2015 / TIR 1394

… It's not about Event Handling, Data Flow
or Graph Processing

It's about Systems' Architecture

a manifesto that
Lead to design
Reliable and Scalable Softwares
According to ~~New~~ Needs

Old needs in Telecom & Embedded Systems
And Now in Everything & Everywhere

Advancements in **Hardware**

Multicore CPUs, Variant Devices, IoT, ...

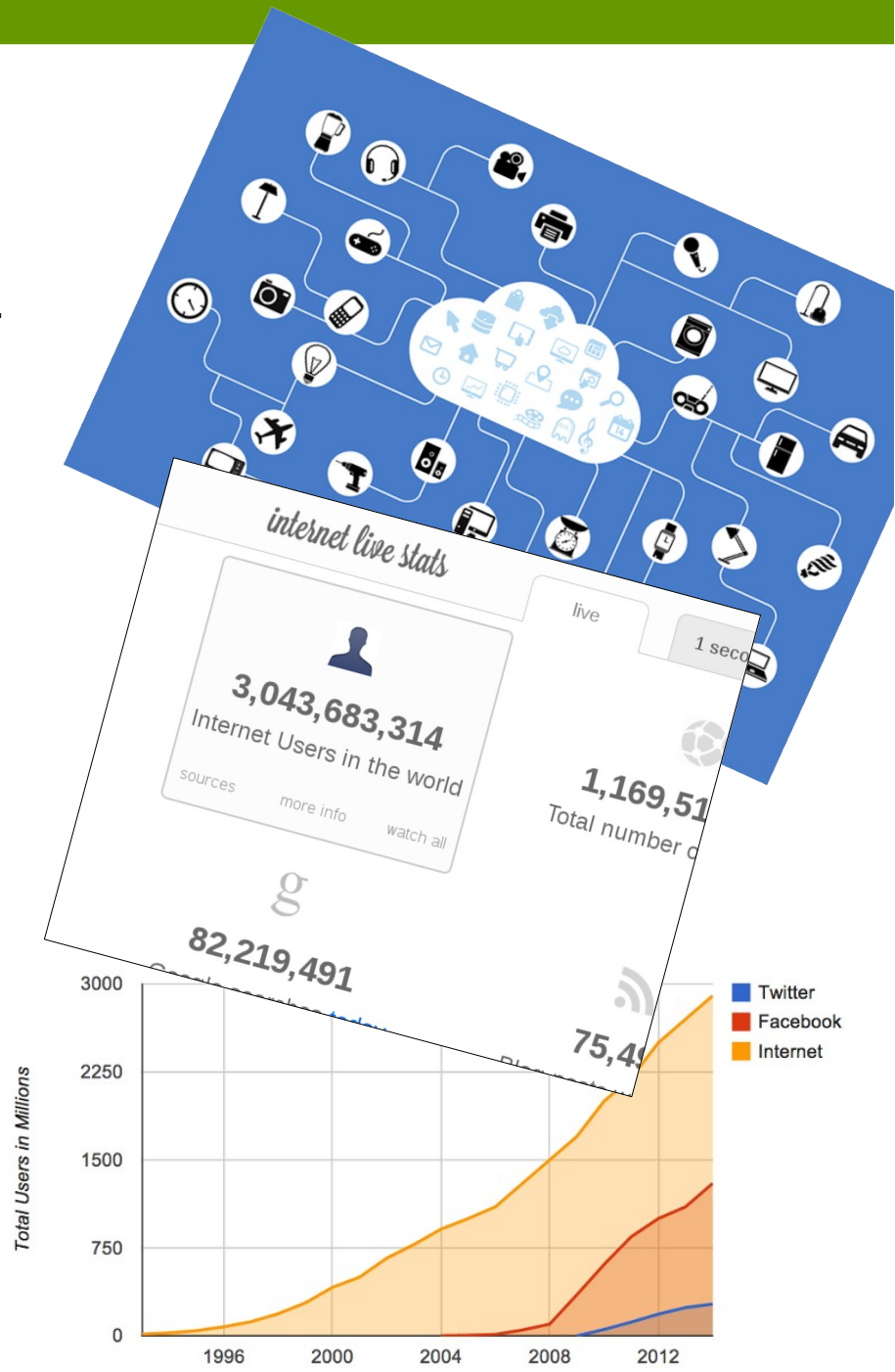
Internet has Grown

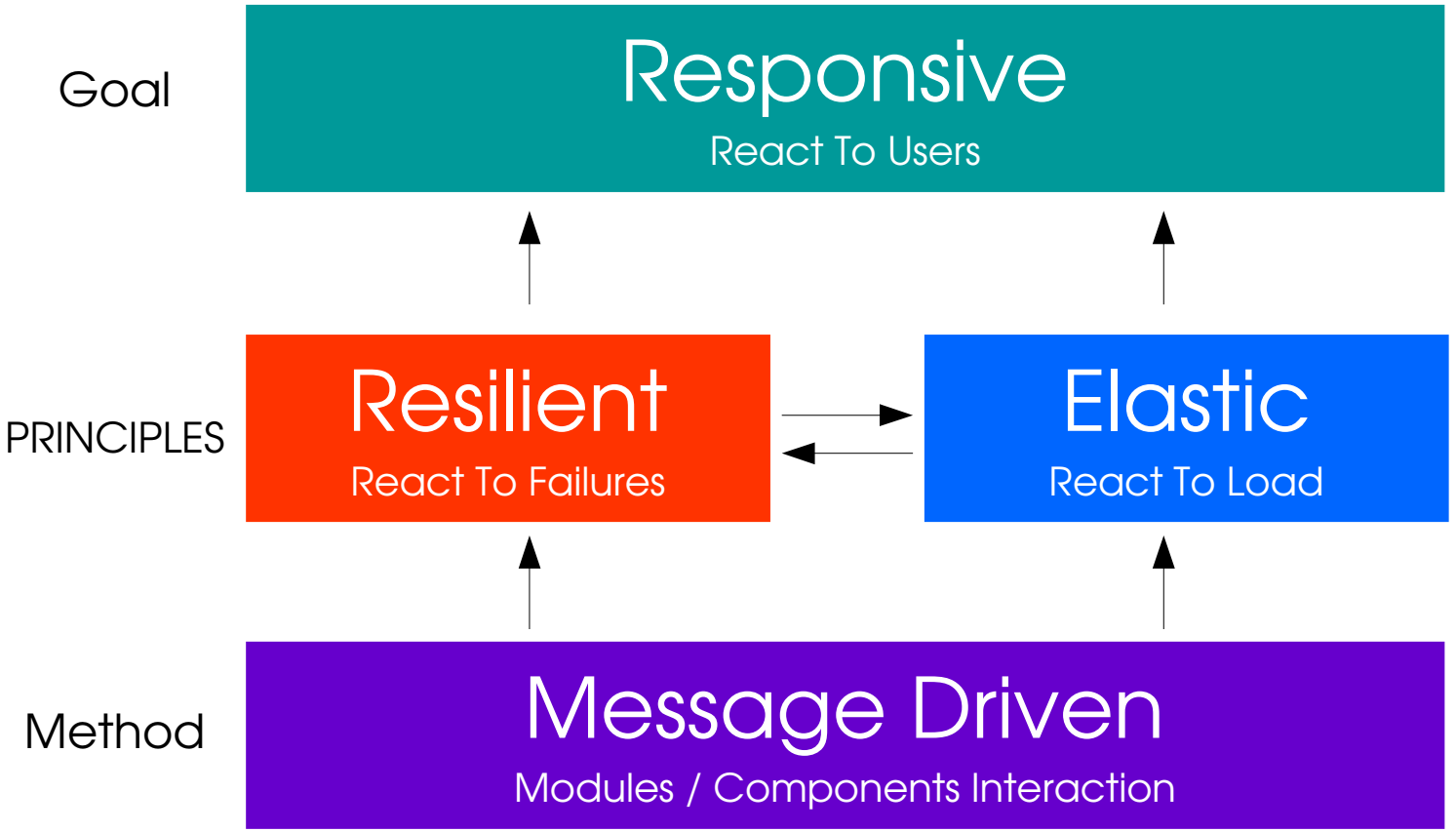
Businnes needs **Users**; And Users:

Don't Like "Slow Responses"

&

Hate "Unavailable Services"




Goal
Responsive
React To Users
PRINCIPLES
Resilient
React To Failures
Elastic
React To Load
Method
Message Driven
Modules / Components Interaction

Responsiveness means

React to Users / Clients in Timely Manner

~= Soft Real-time

As far as they (USERS) know, when the response time exceeds their expectation,
the system is down.

- Release It!

A slow response, on the other hand,
ties up resources in the calling system and the
called system.

- Release It!

Disk IO

Out Of Memory

Memory Page
Swapping

TCP Backlog

Fixed Connection Pool

File Descriptor

Slow Systems aren't Usable

Users Don't Like "Slow Responses"

When You Can't Respond in Timely Manner ...

A Responsive System Depends On
Resilient and Elastic one

Resilient ~= Stability

A Resilient System React to Failures

A resilient system keeps processing transactions, even when there are transient impulses, persistent stresses, or component failures disrupting normal processing. This is what most people mean when they just say stability.

- Release It!

```scala
package enjoyment

import scalikejdbc._
import scala.concurrent.Future

trait ImportantModule{

  def persist[A](rsl:A) = {

    // Database Failure
    DB futureLocalTx { implicit dbsession =>
      val id = sql"INSERT INTO objects VALUES ${rsl}".
        updateAndReturnGeneratedKey.apply

      AnotherExternalService.signal(id)
      /*
        External Service Failure
        Web-Service, Other Storage Service, etc.
       */
    }
  }
}
```

Remember Morphy's Law

Database

Outage

Deadlock

RunTime Exception

Web Service

Starvation

Out Of Memory

Whatever Failure (x_X)

You Can't Test Everything:

Integration, Longevity, Concurrency, ...

Design For Resiliency In Real World

Design For Resiliency
In Real World

- Redundancy (No Single Point Of Failure)
- Supervisor
- Bulk Heads
- Delegating
- Low Coupled Compoenents
- ?

Isolation Over Functionality& Failures +
Abstraction Over Availability / Accessibility with
Message Driven Architecture

Elasticity is about Resources

Resources are Constraint
Then It's Good to
Share **N** Resources *
(on single machine or multiple machines)
Between **M** Applications

* : CPU Cores, Memory, VMs, etc.

In other word Elasticity means Scalability *

Scala Up / Out for Responding to USERS

Scale Down / In for Save COST

An Elastic System can allocate /deallocate resources for every individual component * dynamically to match on demands.

*: Scalability Needs Load Balancing
*: Need To Decoupled Component

Scalability Haiku :
Avoid all shared resources,
But if you can not,
Try to batch, and never block.
(Benjamin Hindman – React 2014 – San Francisco)

And also Abstraction :
Thread vs. Task
Locking vs. Hiding & Proxying

Do Isolation and Abstraction Over
Resources and State
with
Message Driven Architecture

I'm sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea.

The big idea is "Messaging" *

- Alan Kay

Objects are Good for Construct Modules; But let Modules Interact with Messages

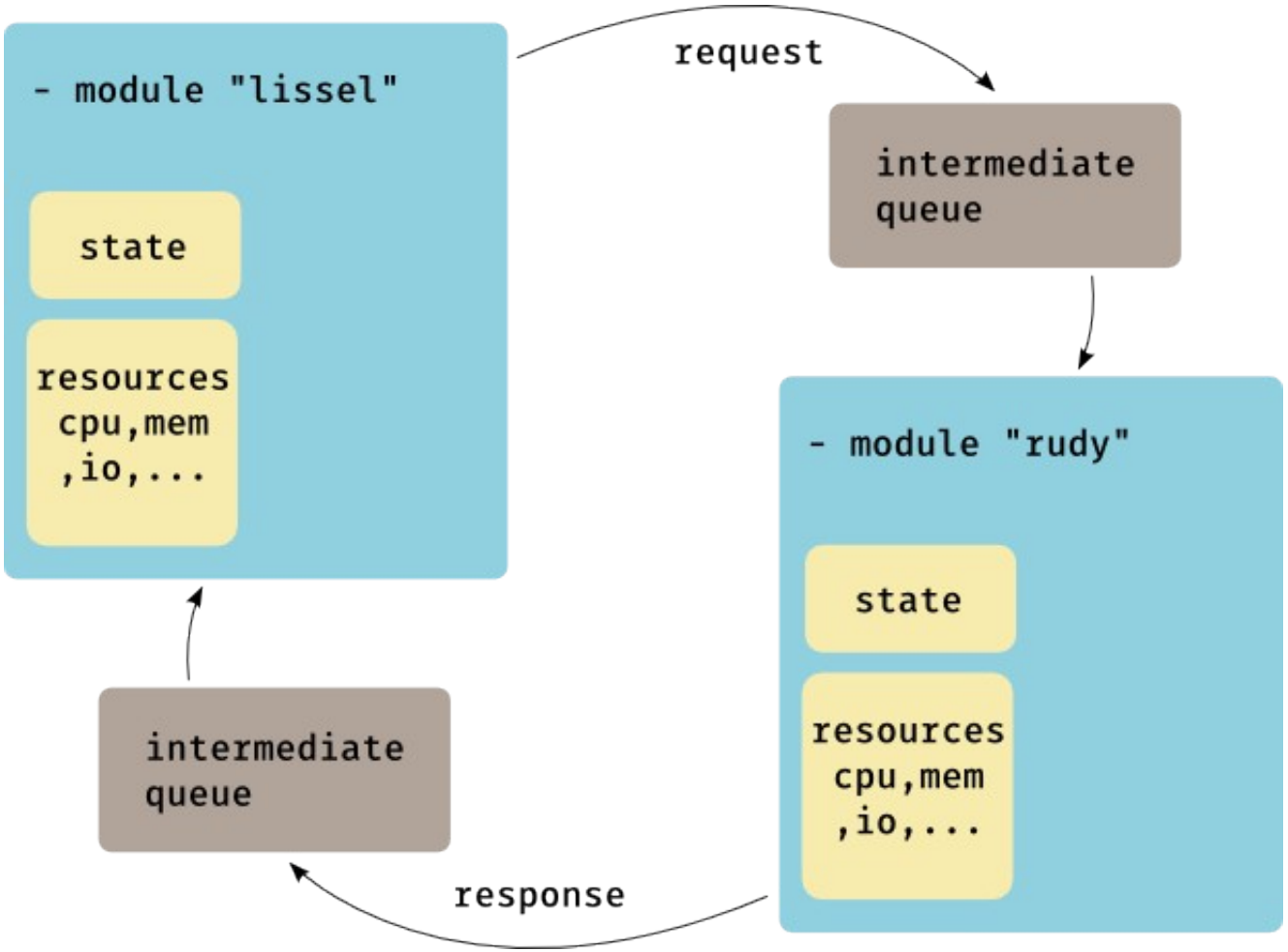
- me

* http://lists.squeakfoundation.org/pipermail/squeak-dev/1998-October/017019.html

Messaging, The Big Idea

vs.

Lock-free Non-Blocking
Share nothing
Location Transparency
Fault Tolerance
Scalable
Better Throughput



module: microservice, actor, ...

Messaging Approaches:

- SOA with Brokers (With Any Language & Any Message Broker)
- Actor Model with Akka & Erlang

And Other Ways:

- CSP with Clojure's core.async & Go's routines
- Event Looping with Vert.x & Node.JS

By Isolation Over Resources / State / Behavior we can achieve Resiliency and Elasticity for Better Throughput and Avg Latency

Your Quality of life after release 1.0 depends on choices you make long before that vital milestone.

- Release It!

- Any Question?

- Thanks

Reza Same'e

Software Developer (these days I use Scala)
Experienced in Web & Reactive

reza.samee@gmail.com
twitter.com/reza_samee
samee.blog.ir