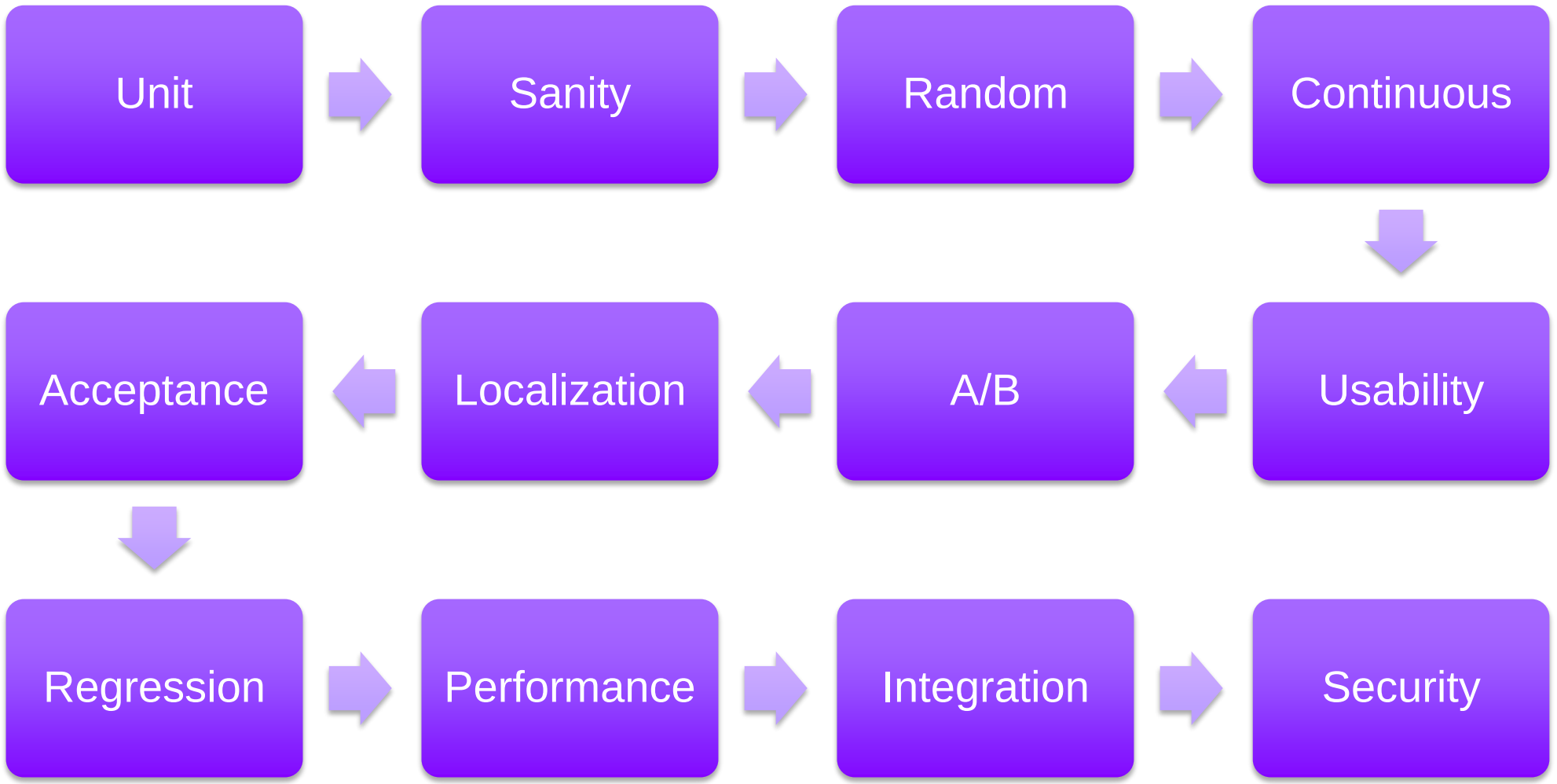
Chaos Engineering at Jet.com

Rachel Reese | @rachelreese | rachelree.se
Jet Technology | @JetTechnology | tech.jet.com

Why do you need chaos testing?

The world is naturally chaotic

But do we need more testing?



You've already tested all your components in multiple ways.

Jet.com

jet.com

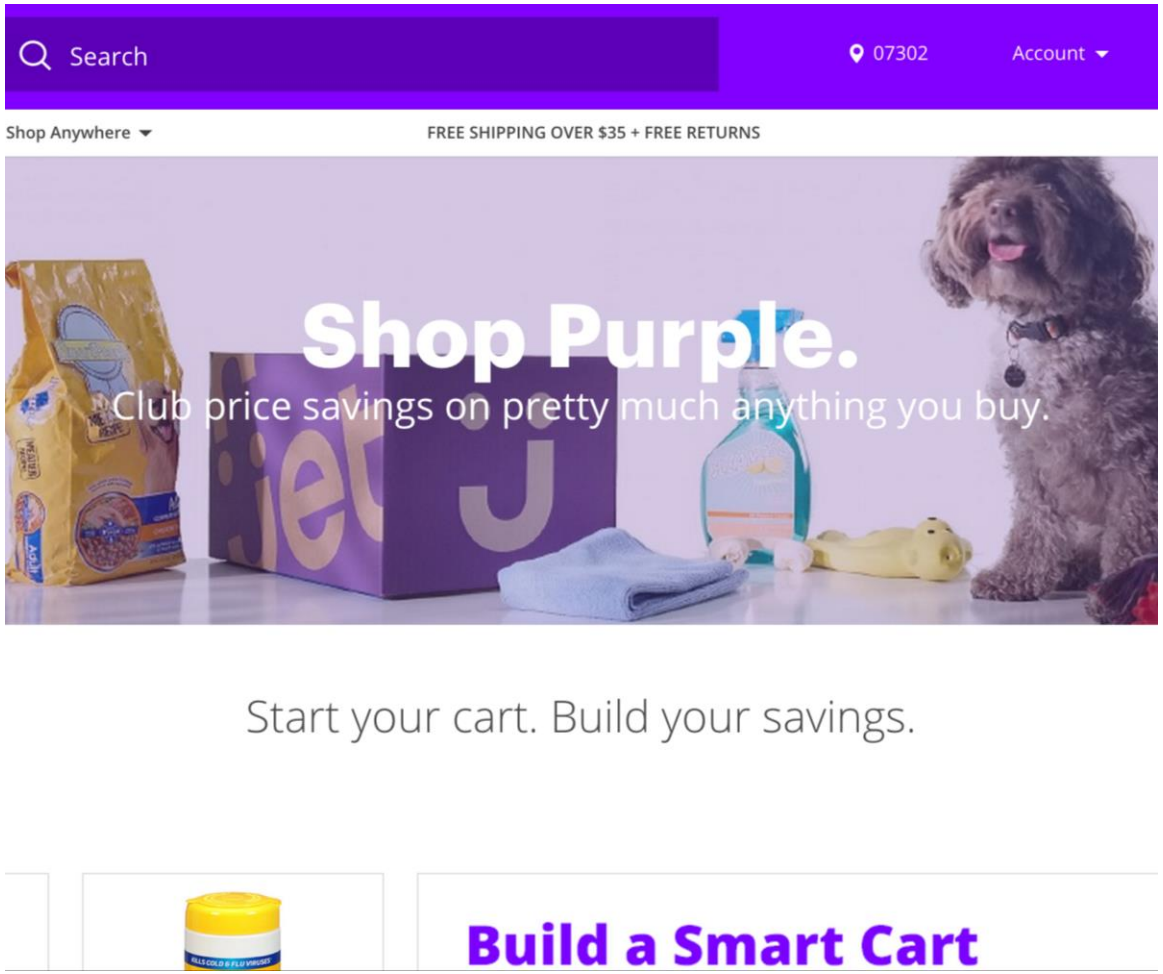
Electronics
TelevisionsLaptopsHeadphones
Need help?
Ask the Jet Heads.
1 (855) 538 4323
help@jet.com
Social
Get our apps
Available on the App Store
GET IT ON Google play
Join the world of Jet

Enter your email address
Join Now

It’s super important to test the interactions in your environment

Jet? Jet who?

Taking on Amazon!



Launched July 22

- Both Apple & Android named our app as one of their tops for 2015
- Over 20k orders per day
- Over 10.5 million SKUs
- #4 marketplace worldwide
- 700 microservices

We're hiring!

http://jet.com/about-us/working-at-jet

Azure	Web sites	Cloud services	VMs	Service bus queues	Services bus topics	Blob storage
Table storage	Queues	Hadoop	DNS	Active directory	SQL Azure	R
F#	Paket	FSharp.Data	Chessie	Unquote	SQLProvider	Python
Deedle	FAKE	FSharp.Async	React	Node	Angular	SAS
Storm	Elastic Search	Xamarin	Microservices	Consul	Kafka	PDW
Splunk	Redis	SQL	Puppet	Jenkins	Apache Hive	Apache Tez

Microservices at Jet

Microservices

- An application of the single responsibility principle at the service level.

> “A class should have one, and only one, reason to change.”

- Has an input, produces an output.

Benefits

Easy scalability
Independent releasability
More even distribution of complexity

What is chaos engineering?

It’s just wreaking havoc with your code for fun, right?

Chaos Engineering is…

Controlled experiments on a distributed system that help you build confidence in the system's ability to tolerate the inevitable failures.

Principles of Chaos Engineering

1. Define “normal”
2. Assume ”normal” will continue in both a control group and an experimental group.
3. Introduce chaos: servers that crash, hard drives that malfunction, network connections that are severed, etc.
4. Look for a difference in behavior between the control group and the experimental group.

Going farther

Build a Hypothesis around Normal Behavior

Vary Real-world Events

Run Experiments in Production

Automate Experiments to Run Continuously

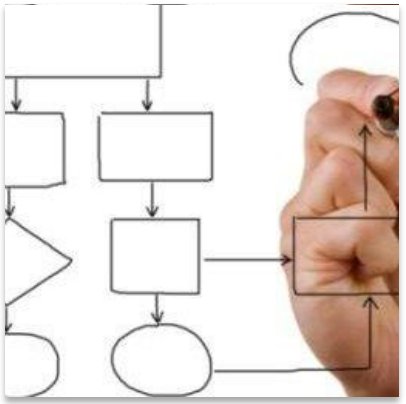
From http://principlesofchaos.org/

Benefits of chaos engineering

Benefits of chaos engineering

You're awake

Design for failure

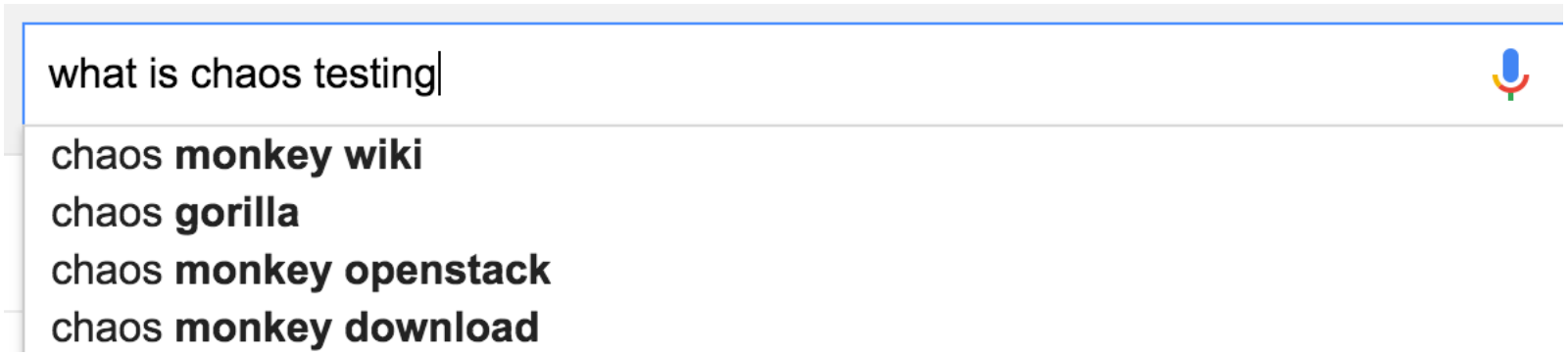


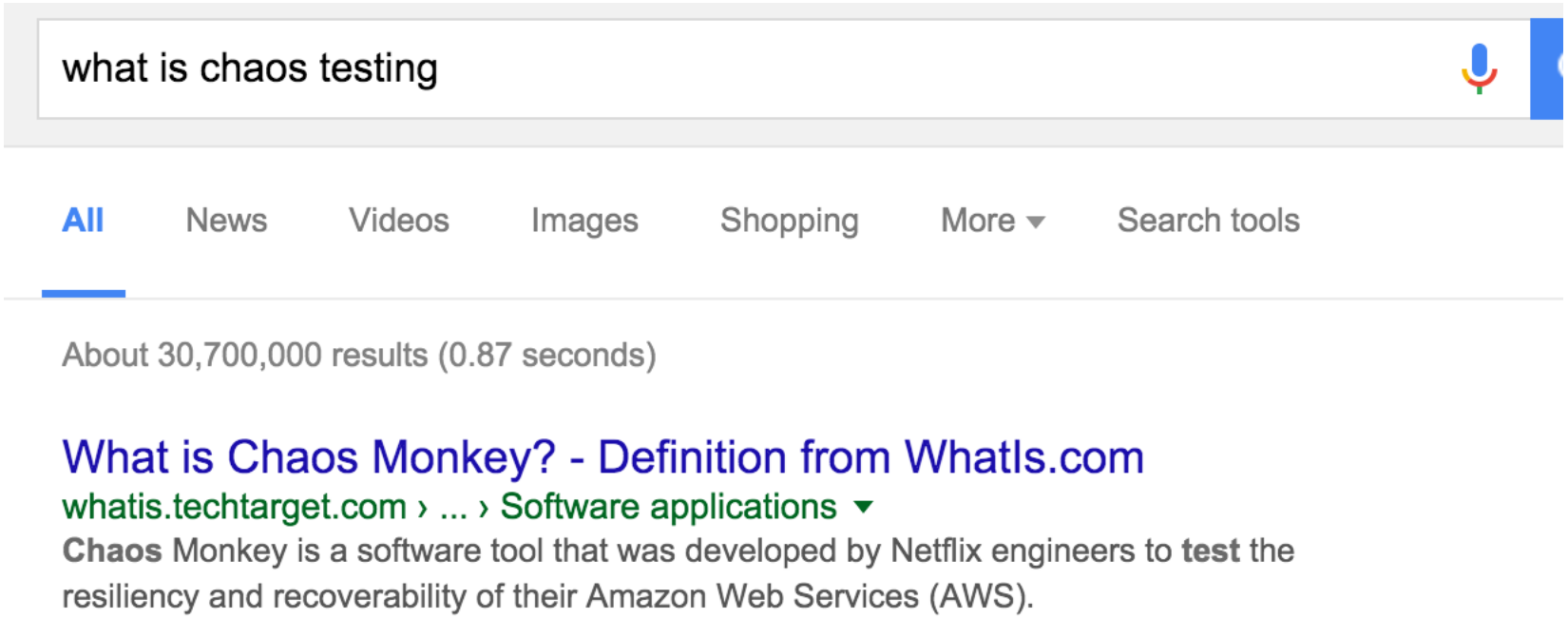
Healthy systems

Self service

Current examples of chaos engineering

Maybe you meant Netflix's Chaos Monkey?





How is Jet different?

We’re not testing in prod (yet).

SQL restarts & geo-replication

Start

- Checks the source db for write access
- Renames db on destination server (to create a new one)
- Creates a geo-replication in the destination region

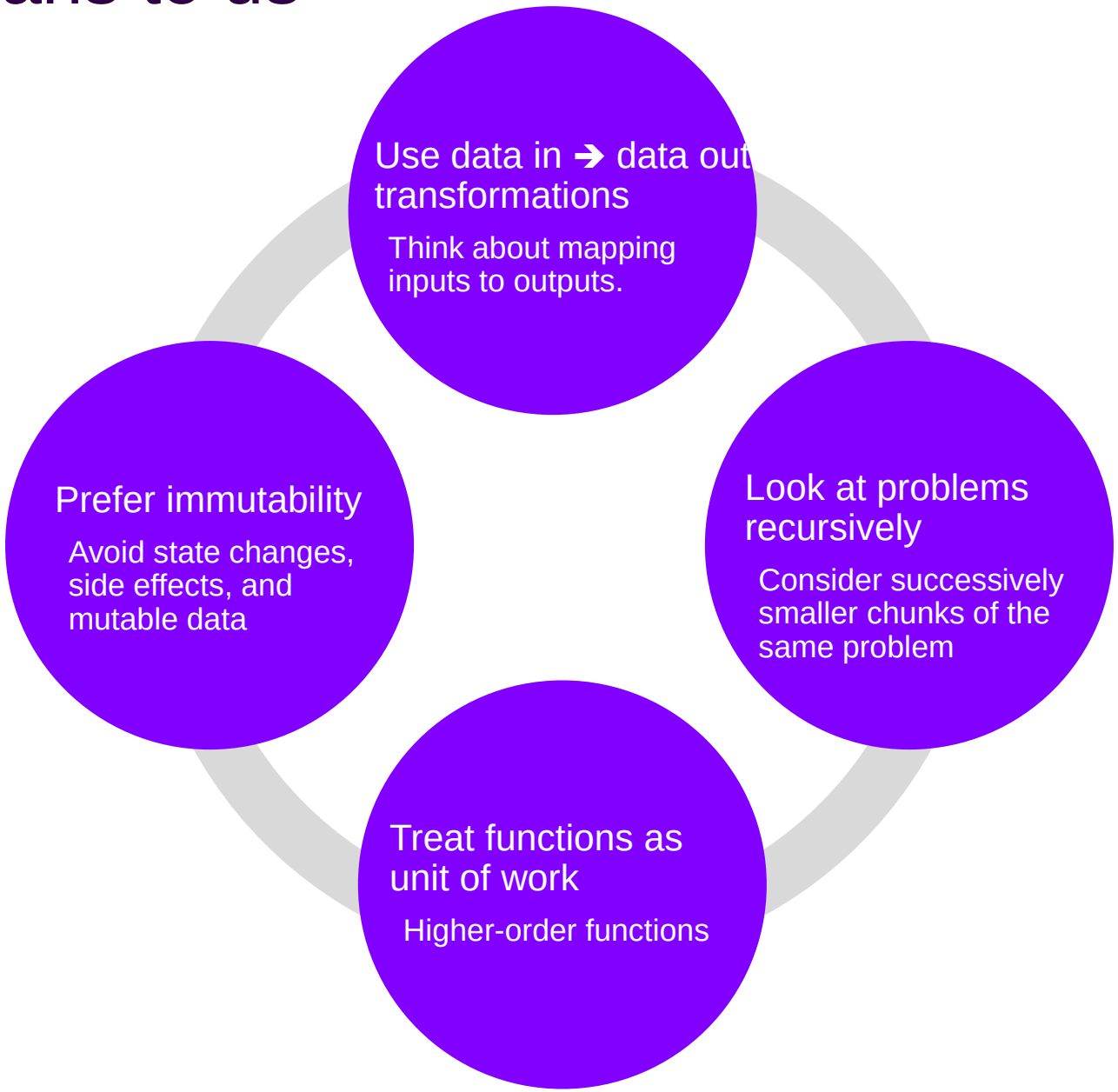
Stop

- Shuts down cloud services writing to source db
- Sets source db as read-only
- Ends continuous copy
- Allows writes to secondary db

Azure & F#

Why F#?

What FP means to us



“The F# solution offers us an order of magnitude increase in productivity and allows one developer to perform the work [of] a team of dedicated developers…”

Yan Cui
Lead Server Engineer, Gamesys

Concise and powerful code

C#

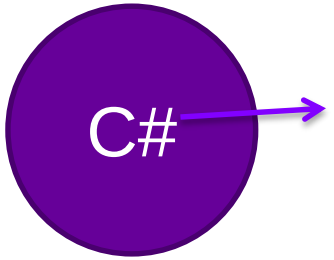
```
public abstract class Transport{  }

public abstract class Car : Transport {
        public string Make { get; private set; }
        public string Model { get; private set; }
        public Car (string make, string model) {
                this.Make = make;
                this.Model = model;
        }
}

public abstract class Bus : Transport {
        public int Route { get; private set; }
        public Bus (int route) {
                this.Route = route;
        }
}

public class Bicycle: Transport {
        public Bicycle() {
        }
}
```

F#

```
type Transport =
    | Car of Make:string * Model:string
    | Bus of Route:int
    | Bicycle
```

Same info as C#!

Trivial to pattern match on!



F# pattern matching

Name	Description	Example
Constant pattern	Any numeric, character, or string literal, an enumeration constant, or a defined literal identifier	`1.0`, `"test"`, `30`, `Color.Red`
Identifier pattern	A case value of a discriminated union, an exception label, or an active pattern case	`Some(x)` `Failure(msg)`
Variable pattern	*identifier*	`a`
as pattern	*pattern* as *identifier*	`(a, b) as tuple1`
OR pattern	*pattern1* | *pattern2*	`([h] | [h; _])`
AND pattern	*pattern1* & *pattern2*	`(a, b) & (_, "test")`
Cons pattern	*identifier* :: *list-identifier*	`h :: t`
List pattern	[*pattern_1*; ... ; *pattern_n*]	`[ a; b; c ]`
Array pattern	[| *pattern_1*; ..; *pattern_n*]	`[| a; b; c |]`
Parenthesized pattern	(*pattern*)	`( a )`
Tuple pattern	(*pattern_1*, ... , *pattern_n*)	`( a, b )`
Record pattern	{ *identifier1* = *pattern_1*; ... ; *identifier_n* = *pattern_n* }	`{ Name = name; }`
Wildcard pattern	_	`_`
Pattern together with type annotation	*pattern* : *type*	`a : int`
Type test pattern	:? *type* [as *identifier*]	`:? System.DateTime as dt`
Null pattern	null	`null`

Concise and powerful code

C#

```
public abstract class Transport{  }

public abstract class Car : Transport {
        public string Make { get; private set; }
        public string Model { get; private set; }
        public Car (string make, string model) {
                this.Make = make;
                this.Model = model;
        }
}

public abstract class Bus : Transport {
        public int Route { get; private set; }
        public Bus (int route) {
                this.Route = route;
        }
}

public class Bicycle: Transport {
        public Bicycle() {
        }
}
```

F#

```
type Transport =
    | Car of Make:string * Model:string
    | Bus of Route:int
    | Bicycle
    | Train of Line:int


let getThereVia (transport:Transport) =
    match transport with
      | Car (make,model) -> ...
      | Bus route -> ...
      | Bicycle -> ...
```

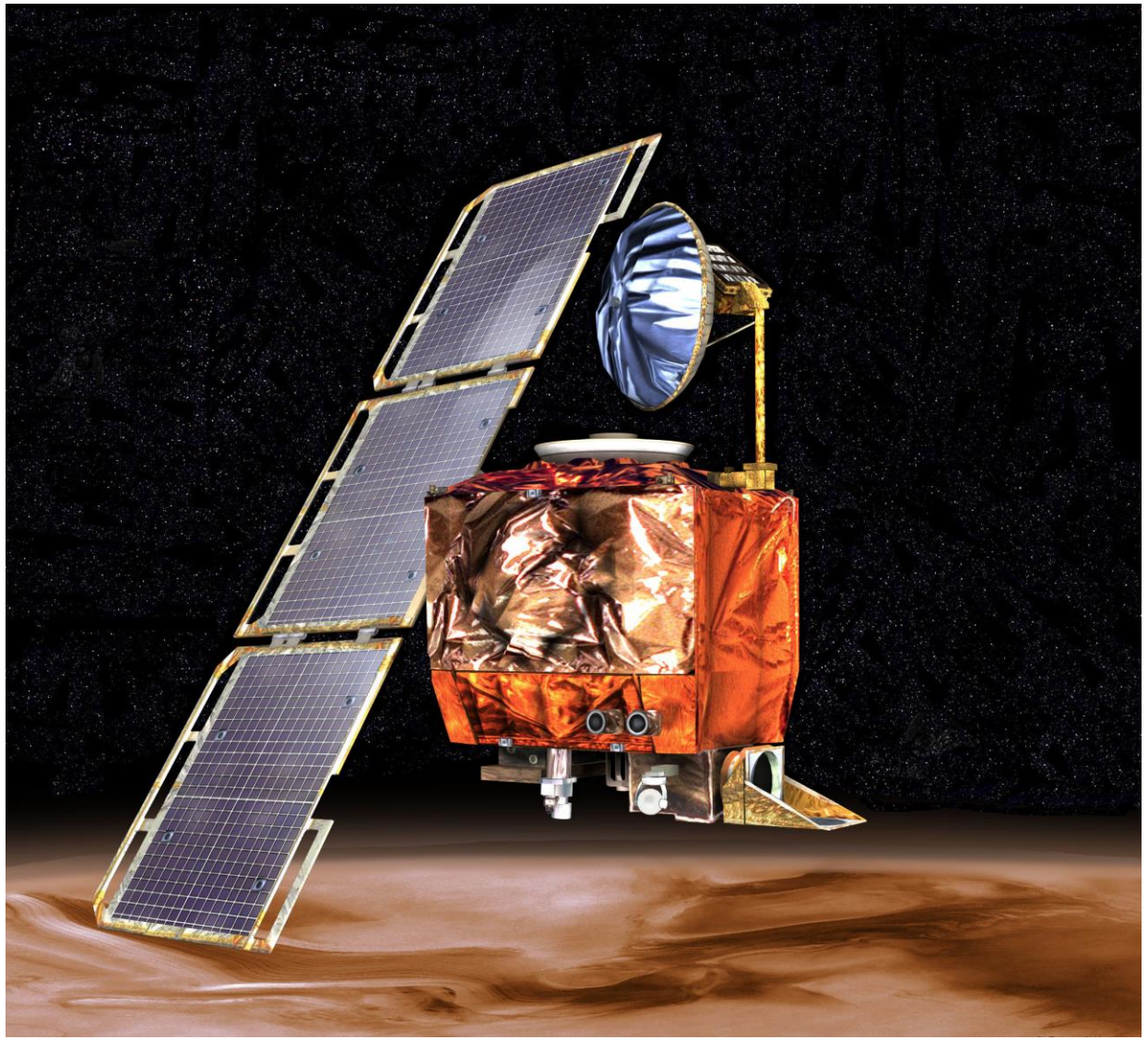
Warning FS0025: Incomplete pattern matches on this expression. For example, the value 'Train' may indicate a case not covered by the pattern(s)

Units of Measure

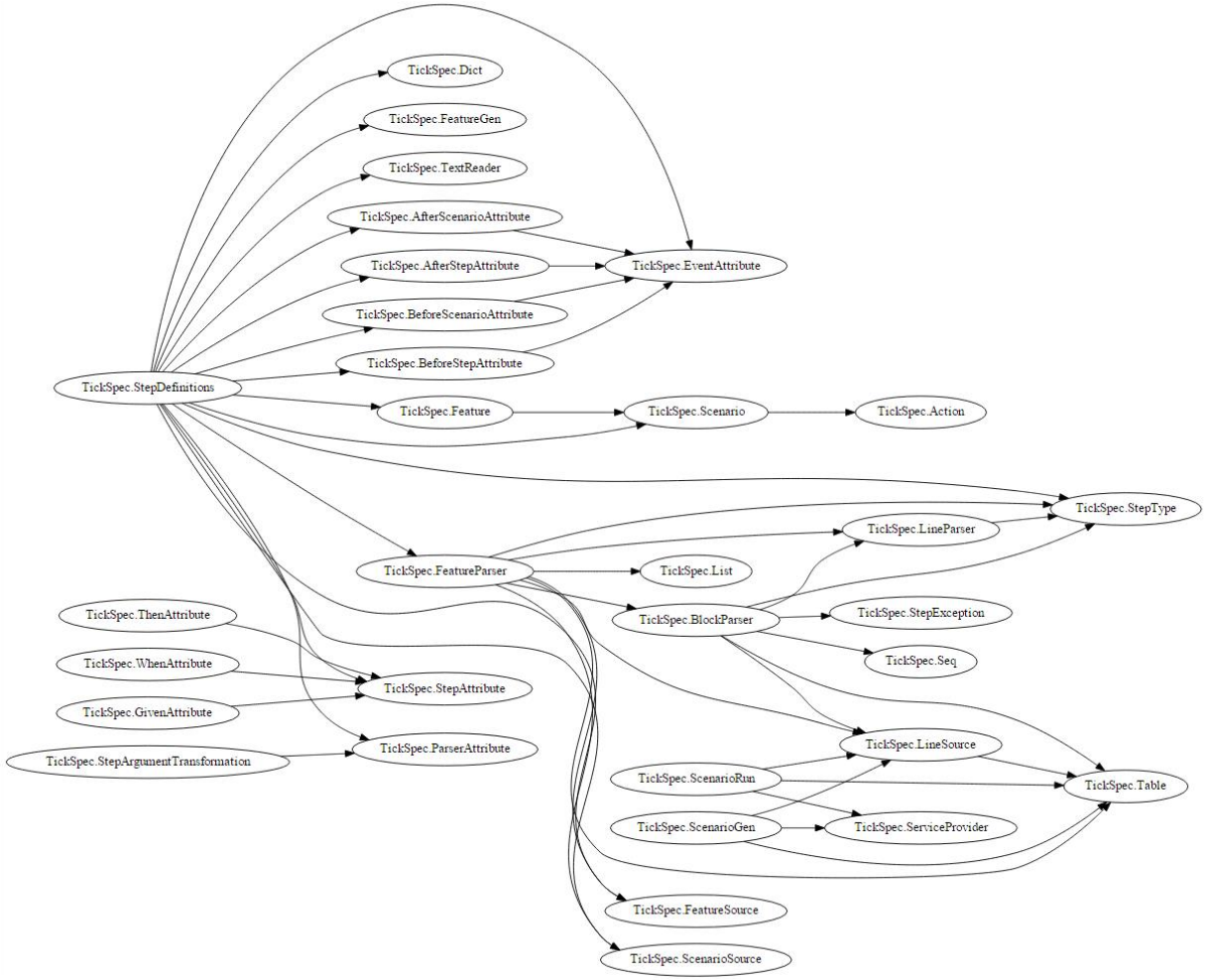
Mystery of Orbiter Crash Solved

By Kathy Sawyer
Washington Post Staff Writer
Friday, October 1, 1999; Page A1

NASA's Mars Climate Orbiter was lost in space last week because engineers failed to make a simple conversion from English units to metric, an embarrassing lapse that sent the $125 million craft fatally close to the Martian surface, investigators said yesterday.



TickSpec – an F# project



Thanks to Scott Wlaschin for his post, [Cycles and modularity in the wild]

SpecFlow– a comparable C# project

Thanks to Scott Wlaschin for his post, Cycles and modularity in the wild

Chaos code!

What do our services look like?

Define inputs & outputs

```
type Input =
  | Product of Product

type Output =
  | ProductPriceNile of Product * decimal
  | ProductPriceCheckFailed of PriceCheckFailed
```

Define how input transforms to output

```
let handle (input:Input) =
    async {
        return Some(ProductPriceNile({Sku="343434"; ProductId = 17; ProductDescription = "My amazing product"; CostPer=1.96M}, 3.96M))
    }
```

Define what to do with output

```
let interpret id output =
    match output with
    | Some (Output.ProductPriceNile (e, price))  -> async {()} // write to event store
    | Some (Output.ProductPriceCheckFailed e) -> async {()} // log failure
    | None -> async.Return ()
```

Read events, handle, & interpret

```
let consume = EventStoreQueue.consume (decodeT Input.Product) handle interpret
```

Our code!

```
let selectRandomInstance compute hostedService = async {
   try
      let! details = getHostedServiceDetails compute hostedService.ServiceName
      let deployment = getProductionDeployment details

      let instance = deployment.RoleInstances
                     |> Seq.toArray
                     |> randomPick

      return details.ServiceName, deployment.Name, instance
   with e ->
      log.error "Failed selecting random instance\n%A" e
      reraise e
}
```

Our code!

```
let restartRandomInstance compute hostedService = async {
   try
      let! serviceName, deploymentId, roleInstance =
         selectRandomInstance compute hostedService
      match roleInstance.PowerState with
      | RoleInstancePowerState.Stopped ->
          log.info "Service=%s Instance=%s is stopped...ignoring...”
             serviceName roleInstance.InstanceName

      | _ ->
         do! restartInstance compute serviceName deploymentId roleInstance.InstanceName
   with e ->
      log.error "%s" e.Message
}
```

Our code!

```
compute
|> getHostedServices
|> Seq.filter ignoreList
|> knuthShuffle
|> Seq.distinctBy (fun a -> a.ServiceName)
|> Seq.map (fun hostedService -> async {
   try
      return! restartRandomInstance compute hostedService
   with
      e -> log.warn "failed: service=%s . %A" hostedService.ServiceName e
      return ()
})
|> Async.ParallelIgnore 1
|> Async.RunSynchronously
```

Has it helped?

Elasticsearch restart

Additional chaos finds

- Redis
- Checkpointing

Jet.com

jet.com

Electronics
TelevisionsLaptopsHeadphones
Need help?
Ask the Jet Heads.
1 (855) 538 4323
help@jet.com
Social
Get our apps
Available on the App Store
GET IT ON Google play
Join the world of Jet

Enter your email address
Join Now

If availability matters, you should be testing for it.

Azure + F# + Chaos = <3

Chaos Engineering at Jet.com

Rachel Reese | @rachelreese | rachelree.se
Jet Technology | @JetTechnology | tech.jet.com
Nora Jones | @nora_js