How to write Cynical software

Stability patterns and anti-patterns

dagi@goodata.com
https://twitter.com/_dagi

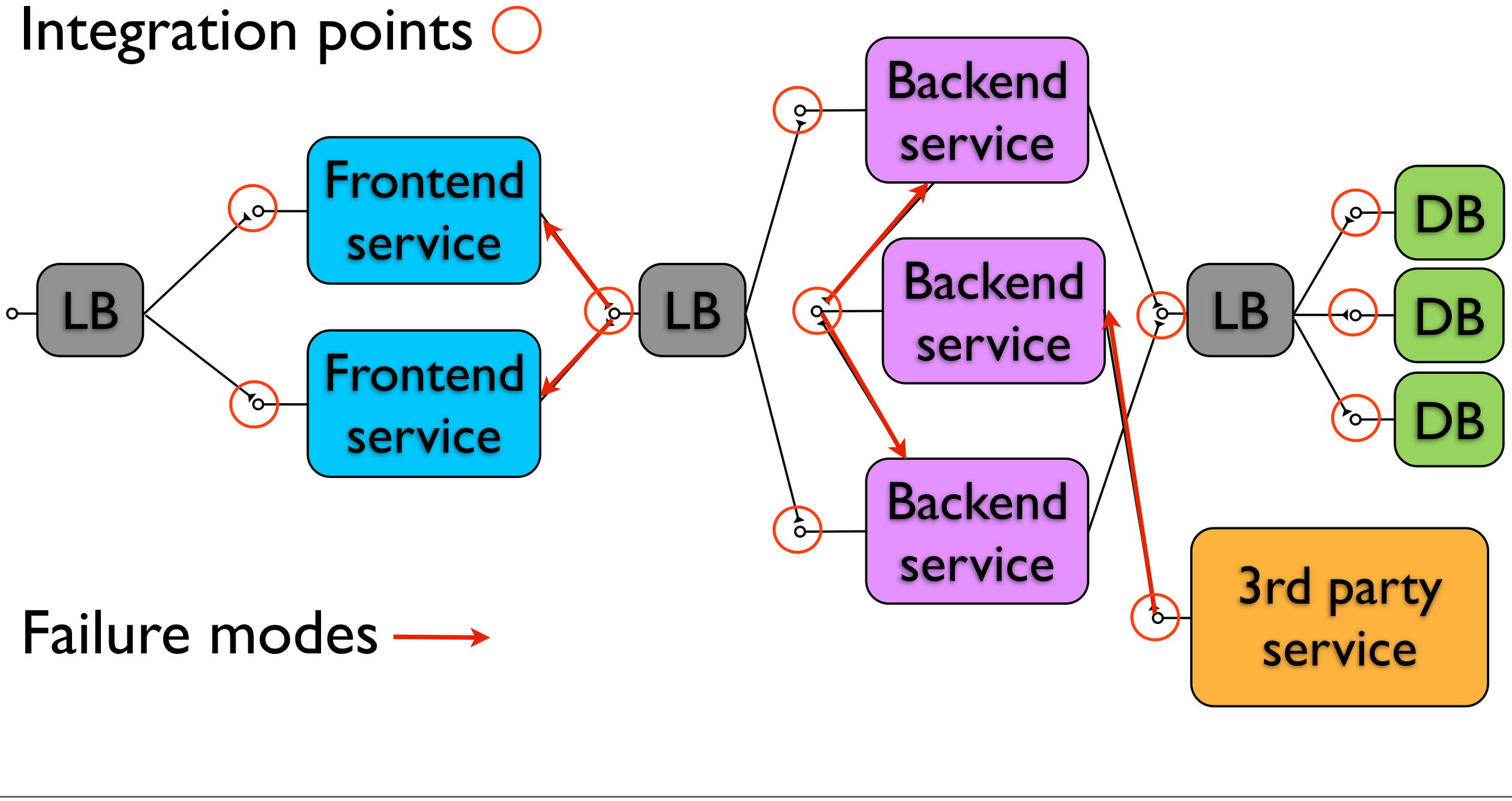
How to become cynical

- Eat your own dog food
  - Strong feedback
  - Pager duty (Engineer on duty)
  - DevOps

How i’ve got there
No throw over the wall syndrom

Monday 4 November 13

Integration points & Failure modes



- shift toward SOA, interconnected services, remote communication, 3rd party services
- how a crack may appear – failure mode
- integration point
- cracks are tightly coupled to integration points
- the way cracks appear and propagated across multiple layers and services
- Integration points may accelerate (chained reaction) or stop cracks
- Failure in one component increase the probability of failure in another component service
- Slow responses, endpoint unreachability
- High levels of complexity provide more directions for the cracks to propagate in.

GoodData

upload_ait_1378821998564 | Support | Zdeněk Koštál

Account
Data Loading Admin
Logout

Upload a **CSV file**

Select a file to upload to your GoodData project.

Choose File...

Uploading tips:

See our CSV formatting guidelines to ensure successful upload

Contact Support if you have any troubles

```
User user; Projects proje
try {
 projects = dao.getProjects(user, OFFSET, LIMIT).await(250, MS);
} catch (FutureTimeoutException e) {
 throw new AccountInfoUnavailableException();
}

for (Project project : projects) {
 Set<String> userPermissions = dao.getPermissions(project, user);
 if (userPermissions.contains(CAN_INIT_
  return new AccountInfo(Boolean.TRUE);
 }
}
```

1

2

Hanging SQL connection (90 s timeout)

Hanging HTTP worker thread (120s)

Hanging HTTP connection (60s timeout)

Cli

nd se

nd server

MySQL

MySQL

MySQL

Frontend server

Backend server

C3

MySQL

MySQL

MySQL

Datacenter Rackspace

GoodData

Give the Bear some rest.

Bear's vital systems are undergoing maintenance

At the moment, the application is undergoing maintenance. We take downtime seriously, and we're working to return to service shortly.

For information regarding this outage:
Support Portal
(415) 200-0194

"Cynicism is merely the art of seeing things as they are instead of as they ought to be" [1.]

A cynical software

Lack of trust

No intimacy

Internal barriers

Bad things happen

Resilience to impulse and stress

Explain the main attributes of cynical software

Steady state

Bulkheads

Test harness

Circuit breaker

Timeouts

Handshaking

Fail fast

Decoupling middleware

Stability patterns & antipatterns

Slow responses

Unbalanced capacities

SLA inversion

Unbounded result set

Blocked threads

Attacks of self denial

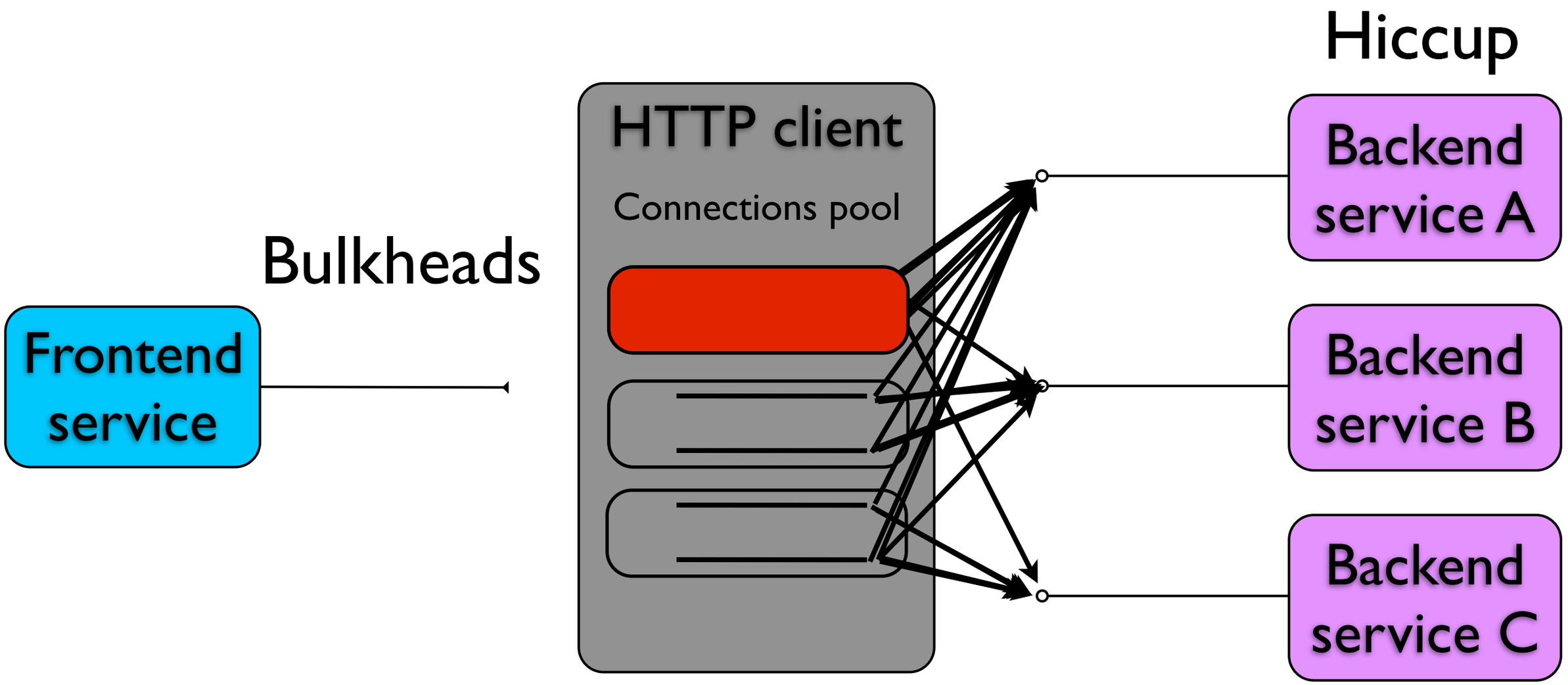
Scaling effects

- the antipatterns will create, accelerate or multiply cracks in the system
- the patterns provide architecture and design guidance to reduce, eliminate, or mitigate the effects of cracks in the system

Circuit breaker

- mediator (decoupling, isolation), integration point wrapper
- fail fast

Bulkheads



fat tails - sizing/capacity
isolation

HYSTRIX

DEFEND YOUR APP

Latency and Fault Tolerance for Distributed Systems

https://github.com/Netflix/Hystrix

- do not reinvent the wheel
- OSS, Java
- most of the patterns are implemented there (circuit breaker, bulkheads, fail fast)

```java
public class CommandHelloWorld extends HystrixCommand<String> {

    private final String name;

    public CommandHelloWorld(String name) {
        super(HystrixCommandGroupKey.Factory.asKey("ExampleGroup"));
        this.name = name;
    }

    @Override
    protected String run() {
        // a real example would do work like a network call here
        return "Hello " + name + "!";
    }
}
```

```java
String s = new CommandHelloWorld("World").execute();

Future<String> fs = new CommandHelloWorld("World").queue();
```


HystrixCommand = circuit breaker + bulkheads
Synchronous/Asynchronous usage
Async usage Future -> timeout

Focus on failures

Testing

Simulate bad things

Chaos monkey [2]

- we do test but on "sligthly" different topology. It makes hard to reveal some kind of bugs
- we mostly test optimistic cases
- HTTP mock server (bad responses, slow responses, protocol violation...)
- Longevity tests

Adaptable design [3]

Architecture

Conway's law [4]

ROC [5]

- early decisions are hard to revert later (costs)
- Big Up Front Design doesn't work prefer Adaptable design - Framework, Platform
- restartability (No restart the world), diagnostics (health checks), recovery mechanism - circuit breaker, isolation/redundancy ()

White box

Black box

Visibility

Operations

Events

Instanemous behavior

Read-Yellow-Green dashboard

- Health checks - do what a user does
- Automatic thread dump on service restart
- White box - logging, Black box - monitoring (JMX)
- Troubleshooting vs. Awareness

The Pragmatic Programmers

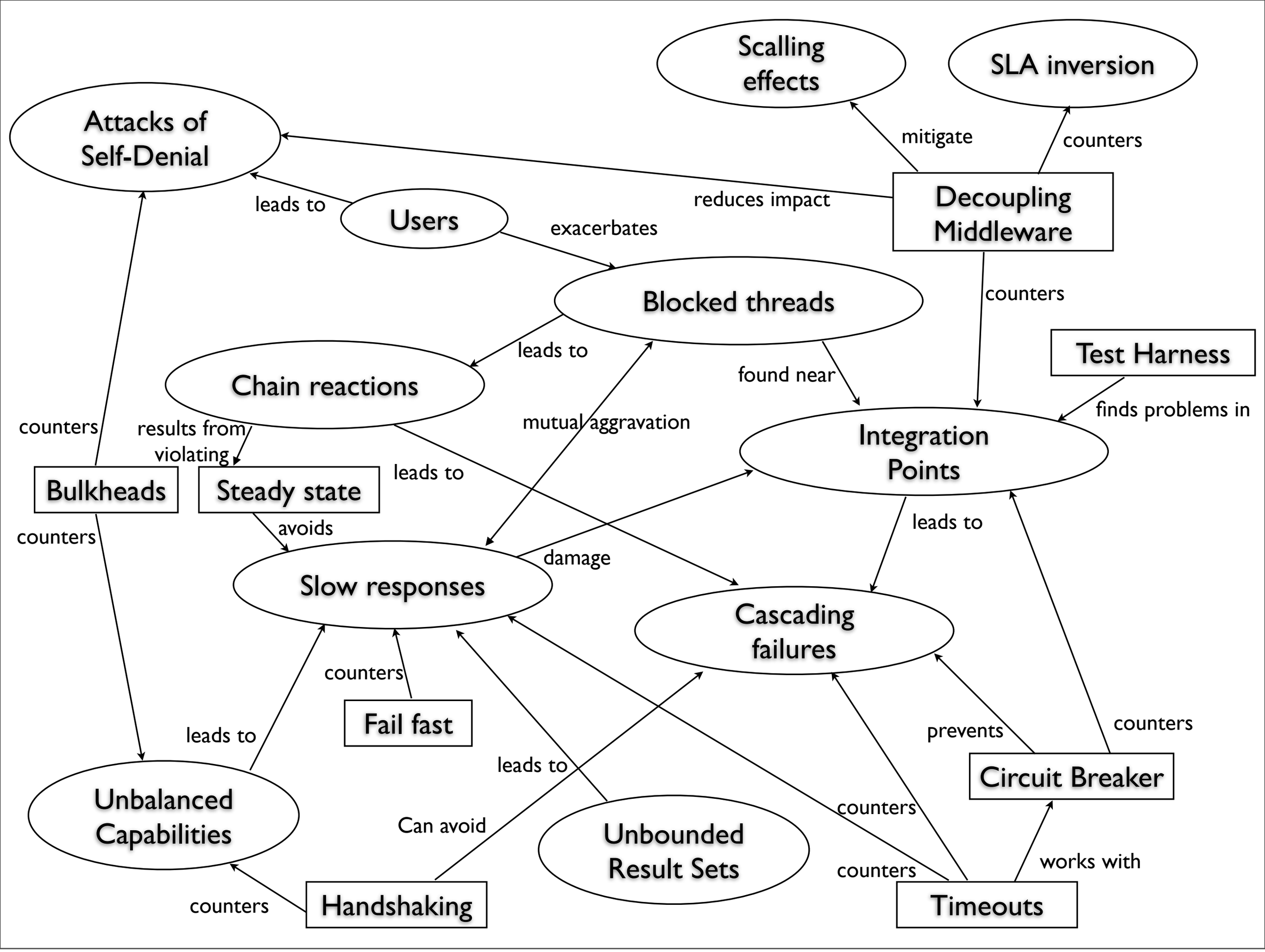
Release It!

Design and Deploy Production-Ready Software

Michael T. Nygard

http://pragprog.com/book/mnee/release-it

Q&A


Scalling effects
SLA inversion
Attacks of Self-Denial
mitigate
counters
reduces impact
Decoupling Middleware
leads to
Users
exacerbates
Blocked threads
counters
leads to
Test Harness
found near
Chain reactions
finds problems in
counters
results from violating
mutual aggravation
Integration Points
Bulkheads
Steady state
leads to
counters
avoids
leads to
damage
Slow responses
Cascading failures
counters
Fail fast
leads to
prevents
counters
leads to
Circuit Breaker
Unbalanced Capabilities
Can avoid
Unbounded Result Sets
counters
counters
works with
counters
Handshaking
Timeouts