

The Limits of Mikado Method

About Me

■ BACKGROUND

Senior developer at 

Started out with Commodore 64 year 1986

Comp. Science programme CTH/GU 2005

A knack for sustainable code, refactoring, ...

<3 Open Source e.g Ubuntu/Linux, Python, Inkscape...



Olof Bjarnason

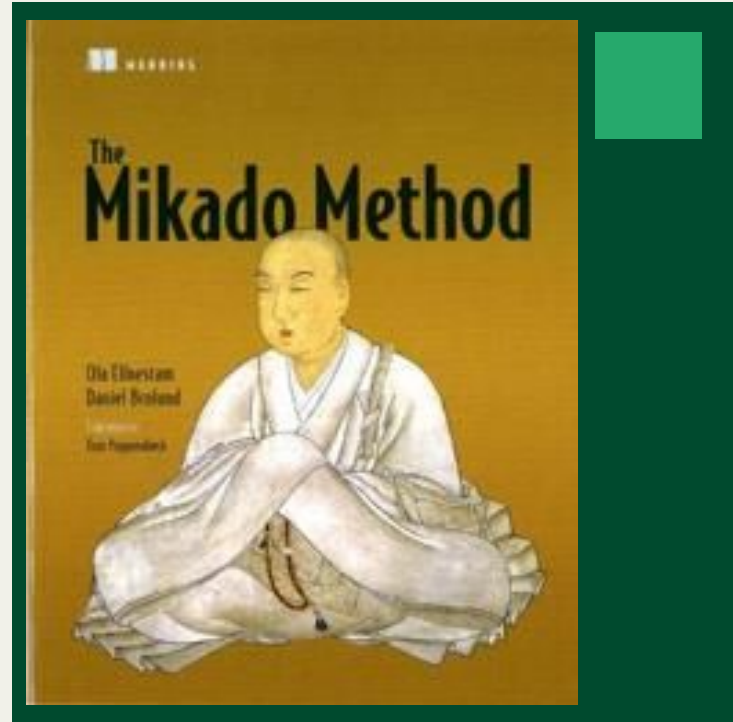
[objarni @ github](#)

olof.bjarnason@proton.me

olofbjarnason.se

The Limits of Mikado Method

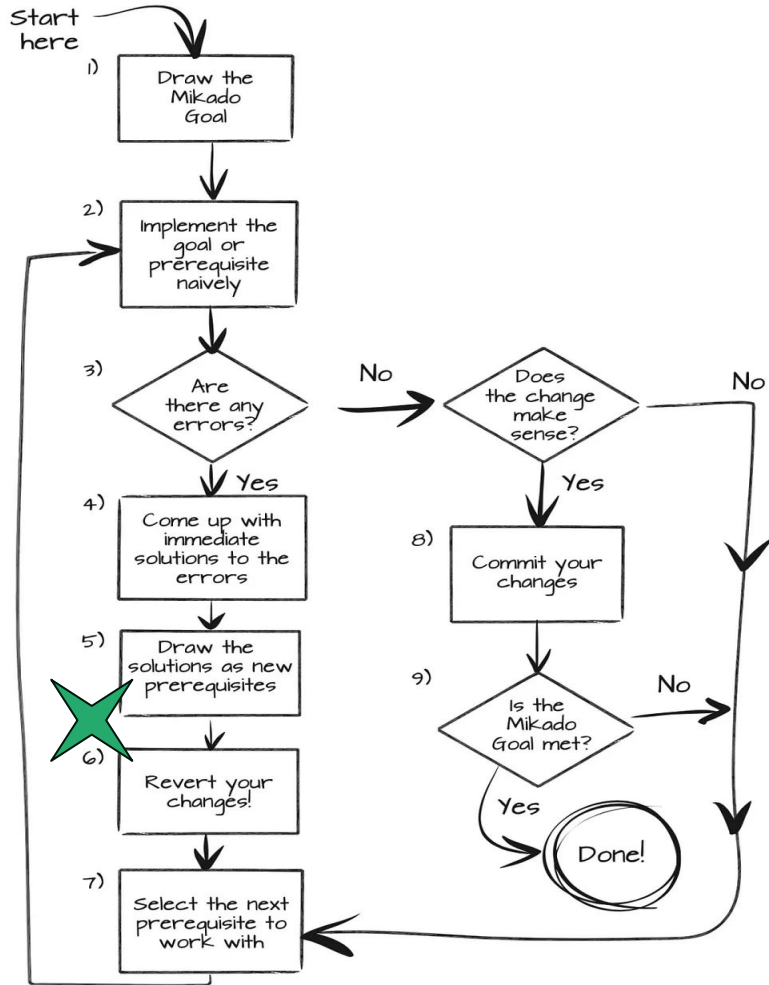
POLL!



Mikado Method?

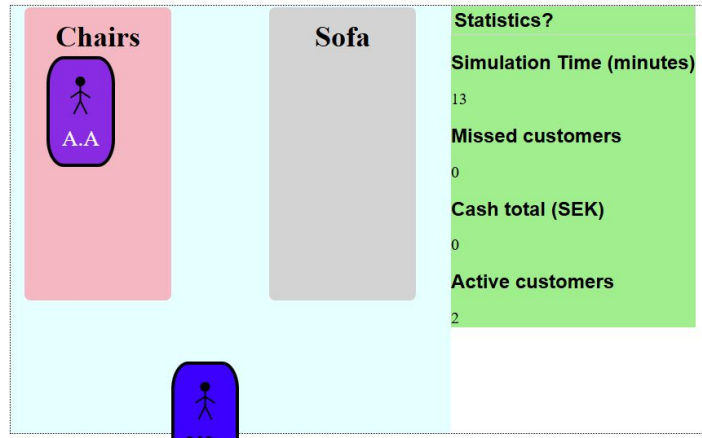
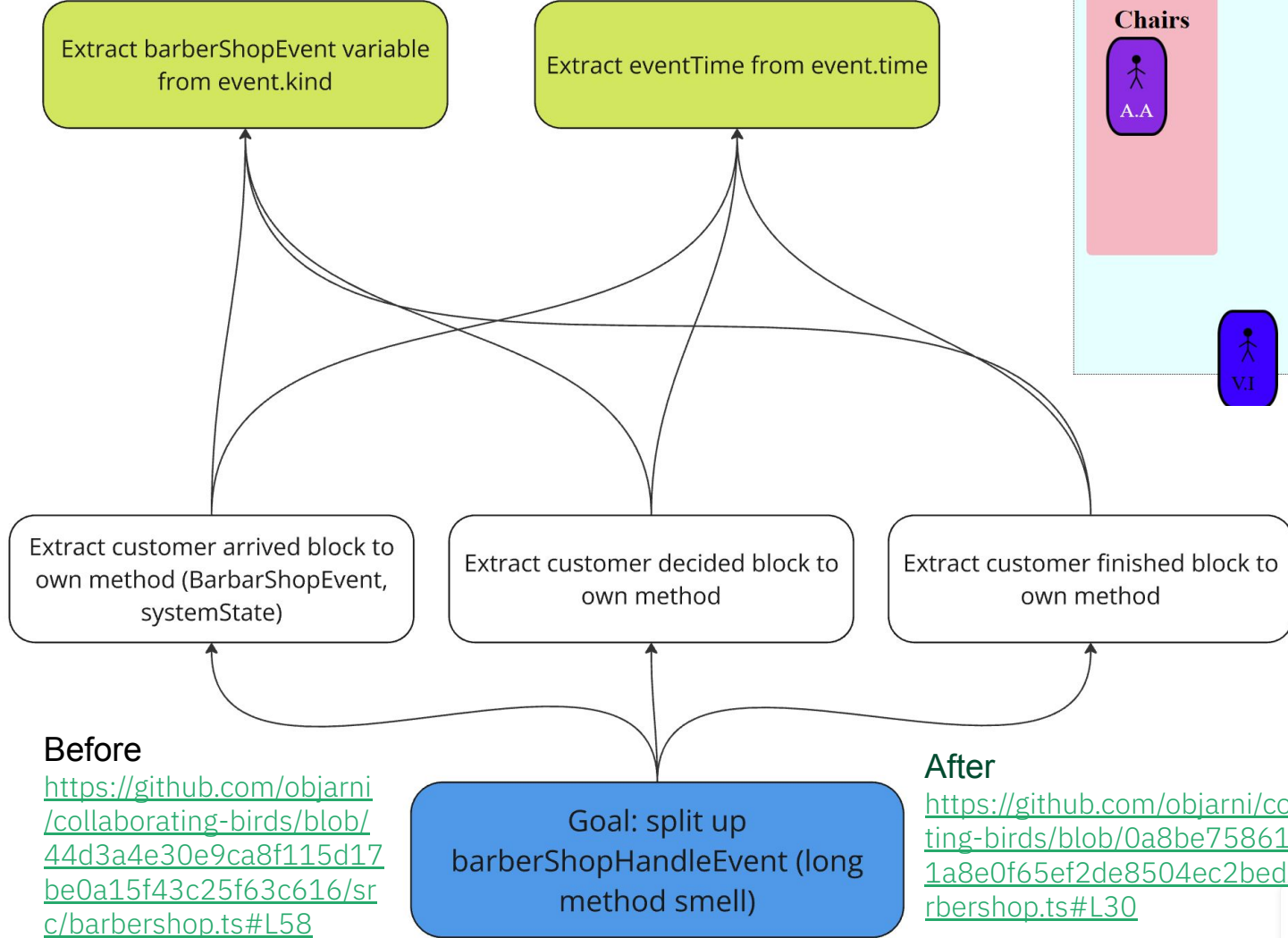
<- “recipe”

(read through 1 min)



[DEMO TIME]





Discrete event simulation
 “Barber shop”

Before

<https://github.com/objarni/collaborating-birds/blob/44d3a4e30e9ca8f115d17be0a15f43c25f63c616/src/barbershop.ts#L58>

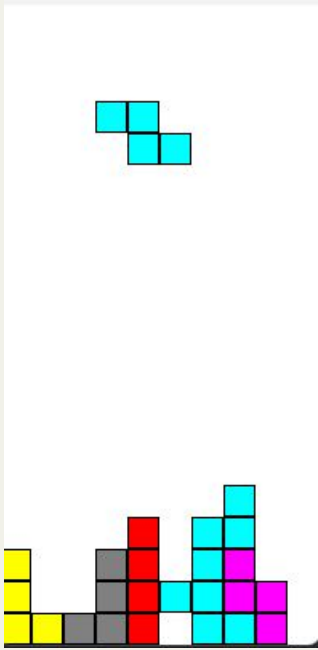
Goal: split up barberShopHandleEvent (long method smell)

After

<https://github.com/objarni/collaborating-birds/blob/Oa8be75861d242871a8e0f65ef2de8504ec2beda/src/barbershop.ts#L30>



Green state test: It builds!



Adding sound effects to a soundless Tetris game



Migration of project to .NET Core

Hydrate IConfiguration at Startup

Use IConfiguration to build configuration for Proposals.Tests.Acceptance

Hydrate IConfiguration instance from appsettings

Add a link to appsettings.Acceptance.json Proposals.Tests.Acceptance

Replace container.AddNewExtension<Bootstrapper>; with container.AddNewExtension(new Bootstrapper(_configuration));

Refactor SQS bootstrapping to this:

```
private void RegisterAmazonSqsClient()
{
    var options = _configuration.GetAWSOptions();
    Container.RegisterInstance(options.CreateServiceClient<IAmazonSQS>());
}
```

Add package AWSSDK.Extensions.NETCore.Setup to load the AWS settings in SchedulerService project and upgrade the AWSSDK.SQS package

Take dependency on IConfiguration in SchedulerService: Bootstrapper & use it to bootstrap

In ProposalGeneratorStarter replace _unityContainer.AddNewExtension<Bootstrapper>() with _unityContainer.AddExtension(new Bootstrapper(_configuration));

Take a dependency on IConfiguration in ProposalGeneratorStarter & populate a local field

From Program.cs, inject the global IConfiguration instance into ProposalGeneratorStarter while setting up TopShell

Remove C:\settings reference from app.config

Copy over the missing settings from app.config to appsettings.*.json files

Update the appsettings.json to include the AWS section like:

```
"AWS": {
  "Profile": "Development",
  "Region": "eu-west-1",
  "ServiceURL": "https://localhost:4576"
}
```

Also add these config values to Octopus

Copy over the development values for all configurations from C:\settings\ProposalGenerator_AppSettings.config to app.config

Use internal Serilog Splunk sink

Install internal package into SchedulerService project

Remove the obsolete Configuration builder method in Bootstrapper

Remove Serilog's Splunk sink (from code and config)

Copy the Bootstrapper:: GetConfigurationSettings() to Program.cs + invoke it from the main() method and store the instance into a static field. Mark the Bootstrapper method Obsolete.

Change the logger configuration in Program.cs to bootstrap the Coolblue sink: using the SplunkViaUdp extension method new SplunkUdpSinkConnectionInfo(Configuration.GetValue<string>("splunk-connection-info: host"), Configuration.GetValue<int>("splunk-connection-info: port")) and remove this line: ReadFrom.AppSettings();



Benefits of Mikado Method

Get unstuck - avoid analysis paralysis

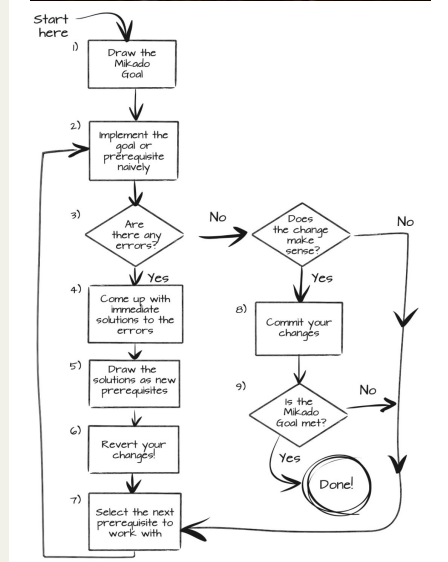
Small commits

Enable Trunk-Based Development

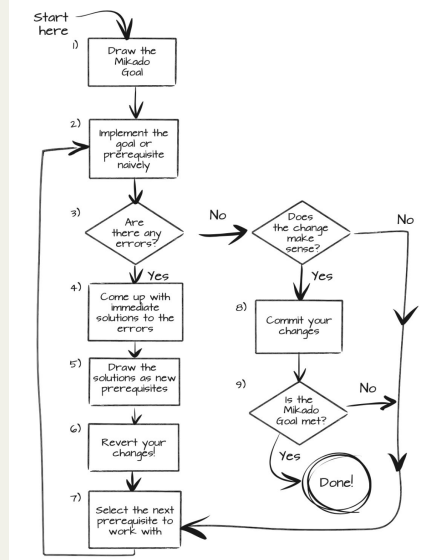
Transparency to team/org “What is going on?” during big change work

“No assumptions - relentlessly pragmatic”

Reduce cognitive load - enable “get back to work with mental model intact - what was I doing again yesterday?”



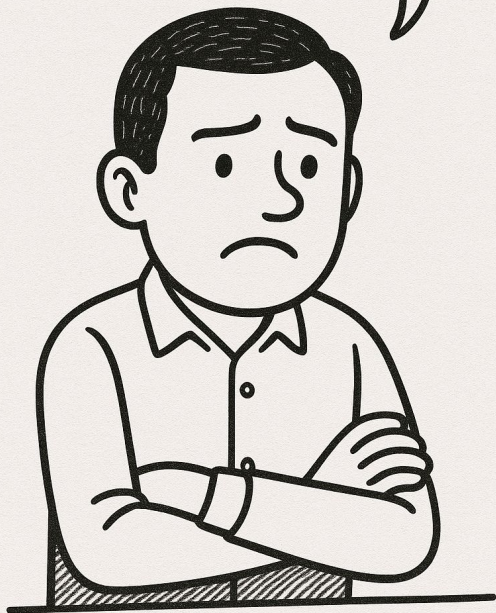
That sounds nice -
what's the catch...?



Limit

#01

THERE ARE
NO TESTS



THERE ARE
NO TESTS



.. well, then
add tests!

:)

SOLUTION - TRY THIS

Economic hint:
Try **Approval
Tests*** to cover
large swaths of
code in one go

```
ReceiptTest.TestPurchase.recei * ReceiptTe
1 1
2 1 - Candy Bar @ - $0.50
3 2 - Soda @ $1.00 = - $2.00
4 - - - Subtotal = - $2.50
5 - - - Tax (10%) = - $0.25
6 - - - Total = - $2.75
```

* also known as characterization tests, golden images tests or text tests

Limit

#02

THERE ARE RELIABLE
TESTS – BUT THEY TAKE
MORE THAN 1 MINUTE
TO RUN!





May seem fine
first couple of
hours...

SOLUTION - TRY THIS

.. but with time, it
becomes unwieldy.
Cycle time is too long!

Find other ways to
test the system
automatically.

- parallelize test runs
- test subsets
- special tooling
- ... be creative!



Limit

#03

SOLUTION - TRY THIS



Audience
turn! What
todo?

?



(what is the pattern here?)

Yes - cycle
time.



Questions?



FEEDBACK

call to action slide

Thanks for
listening!

