test && commit || revert

TCR

~~./test~~

```
./test && commit || revert
```

Thomas Deniffel

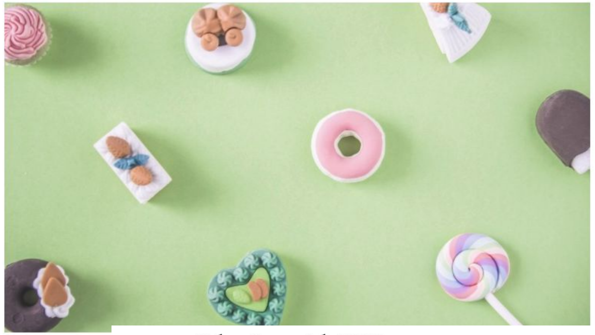
10 Years Programming for Money

TDD

TCR Since October

TCR Variants (test && commit || revert)

Thomas Deniffel
Nov 16, 2018 · 5 min read



In his post, Ke
simplification.
you can imple

Note: TCR is n
context and an

The Origi

TCR Tool (test && commit || revert)

Thomas Deniffel
Nov 29, 2018 · 2 min read

These days I try TCR in different real-world-project to evaluate if it is good or bad. Most of the time I use the variant 'The Relaxed' via a simple Bash-script.

***Note**: TCR (test && commit || revert) is new to you? It is like TDD, but different. Check out this post, which provides background, context and an example.*

Why

The script has two downsides that disturb my typical workflow:

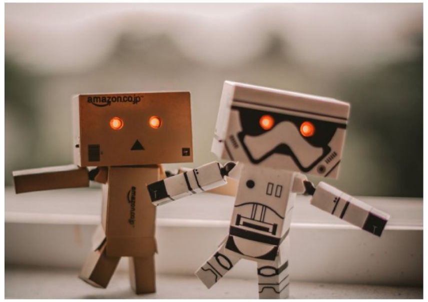
; when I execute it in an

s where, etc.

can see an example

TCR (test && commit || revert). How to use? Alternative to TDD?

Thomas Deniffel
Nov 14, 2018 · 13 min read



`&& commit || revert` (TCR) was introduced by Kent Beck some weeks
Every time you run a test, your code gets either committed or deleted.
has a profound impact on how you develop software and what becomes
ible.

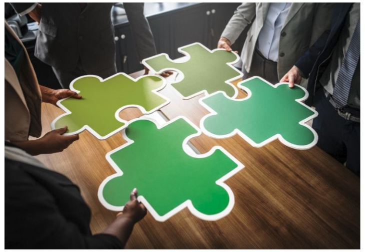
Code-Sync

Thomas Deniffel
Dec 8, 2018 · 3 min read

A while ago, we started to do our mob-programmings remotely in our Meetup-Group (Bavarian Coding Group). After some research and the suggestion, that we could do it via screen-sharing or TeamViewer, we discovered the new (back then) skill of VS Code: Live Code Sharing.

One person opens the project, the others connect, and everyone can type
ile the others see the keystrokes in real-time in their editor. The sharing
rked just fine.

lr *To work together on a code-base synchronized in "real-time"*
chronization can only happen when the code is in a syntactical and
iantical correct state. Tests verify both. So only synchronize when your tests
s. You can use Git together with a bash-script to sync automatically in the
kground.



Fake-Bot with TCR

Thomas Deniffel
Feb 19 · 8 min read

Is it possible to write production-code only by writing tests?

When you do write tests, you declaratively express, what your system should do. When you see TDD as declarative programming (while wearing the test-hat) is it possible to define an engine, that fulfills all the requirements (as a SQL-Query-Engine)?

tl;dr *Fake-Bot automatically makes all your unit-tests green by faking. This allows you to describe your system declaratively through table-based tests. With refactoring, you generalize away from a special solution. TCR helps you stay green.*

First Fake It, Then Make It

When a bot always makes a red test green, we are never in a red state. Refactoring keeps us also in a green state. TCR ensures this.

The idea is, that a bot analyzes the failing tests even before the programmer sees it and makes it green through faking.

Real-World TCR

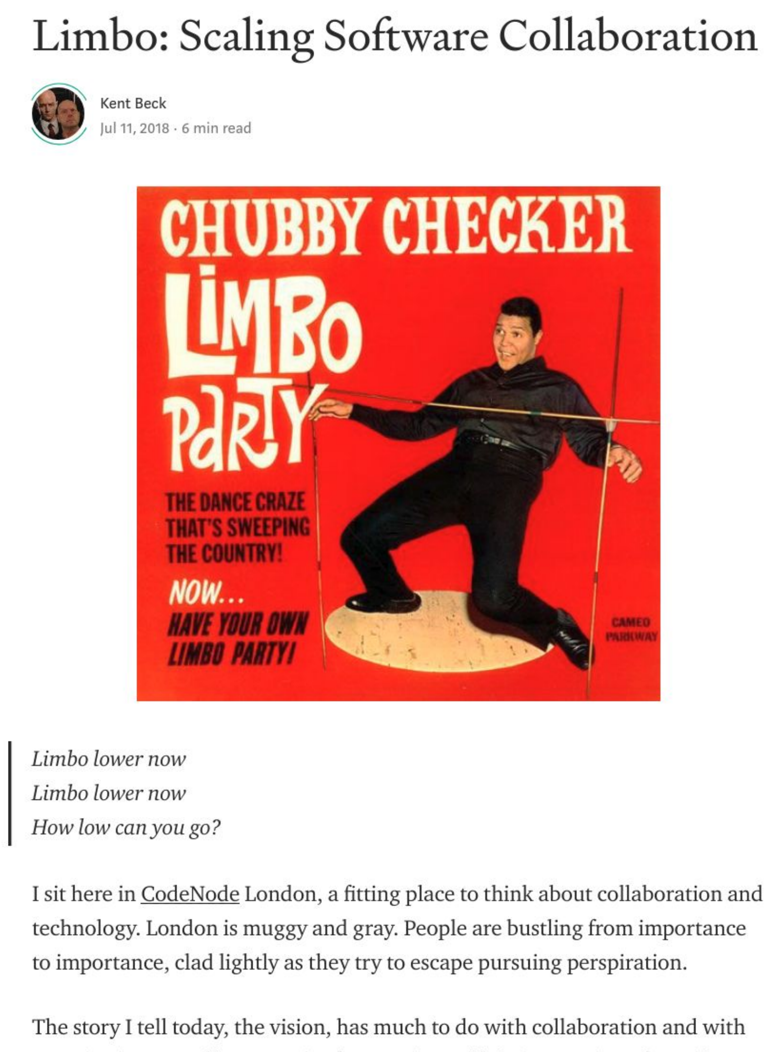
Thomas Deniffel
Feb 21 · 8 min read

When you read about TCR, you always get the Fibonacci-Example or—if you are lucky—something slightly bigger. But you don't find any "real-world" examples. This article tries to fill this gap. It provides an example of a web-app with a web-API for customer and order management done with TCR.

tl;dr *TCR is not exciting, but useful as it will outsource your discipline. Real-World projects are possible without much more effort than TDD through the techniques we already know from TDD.*

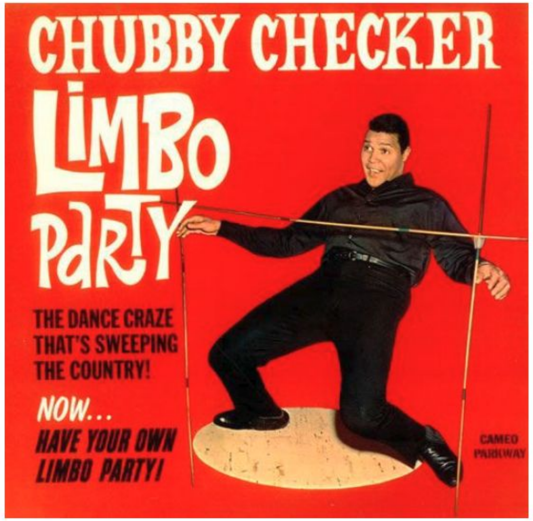
This tutorial is optimized for showing TCR and not to deliver a useful product. Therefore the code quality and structure suffer. But they should be easy to refactor (the test coverage is there through TCR).

06. July 2018 Limbo:



Limbo: Scaling Software Collaboration

Kent Beck
Jul 11, 2018 · 6 min read

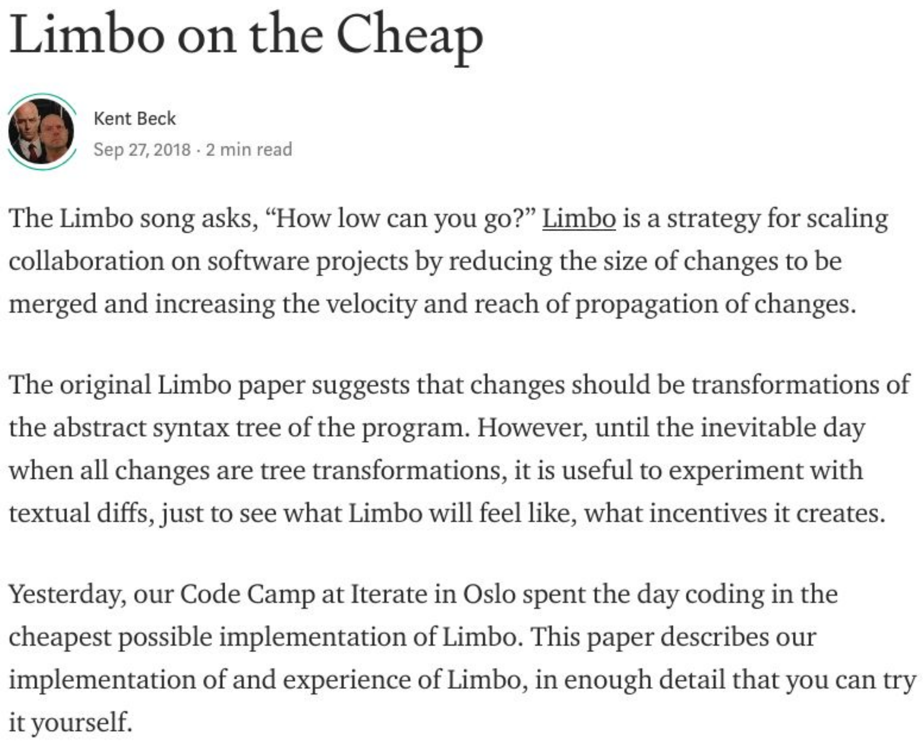

CHUBBY CHECKER
LIMBO PARTY
THE DANCE CRAZE THAT'S SWEEPING THE COUNTRY!
NOW... HAVE YOUR OWN LIMBO PARTY!
CAMEO PARKWAY

> *Limbo lower now*
> *Limbo lower now*
> *How low can you go?*

I sit here in CodeNode London, a fitting place to think about collaboration and technology. London is muggy and gray. People are bustling from importance to importance, clad lightly as they try to escape pursuing perspiration.

The story I tell today, the vision, has much to do with collaboration and with

27.11.2019



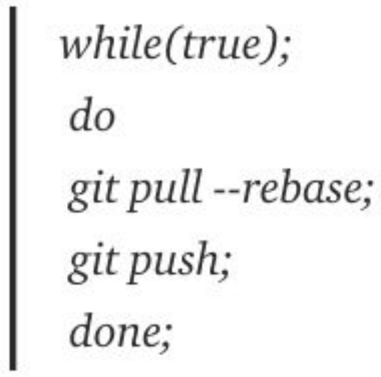
Limbo on the Cheap

Kent Beck
Sep 27, 2018 · 2 min read

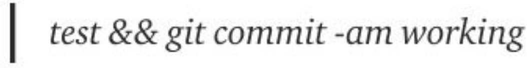
The Limbo song asks, “How low can you go?” Limbo is a strategy for scaling collaboration on software projects by reducing the size of changes to be merged and increasing the velocity and reach of propagation of changes.

The original Limbo paper suggests that changes should be transformations of the abstract syntax tree of the program. However, until the inevitable day when all changes are tree transformations, it is useful to experiment with textual diffs, just to see what Limbo will feel like, what incentives it creates.

Yesterday, our Code Camp at Iterate in Oslo spent the day coding in the cheapest possible implementation of Limbo. This paper describes our implementation of and experience of Limbo, in enough detail that you can try it yourself.

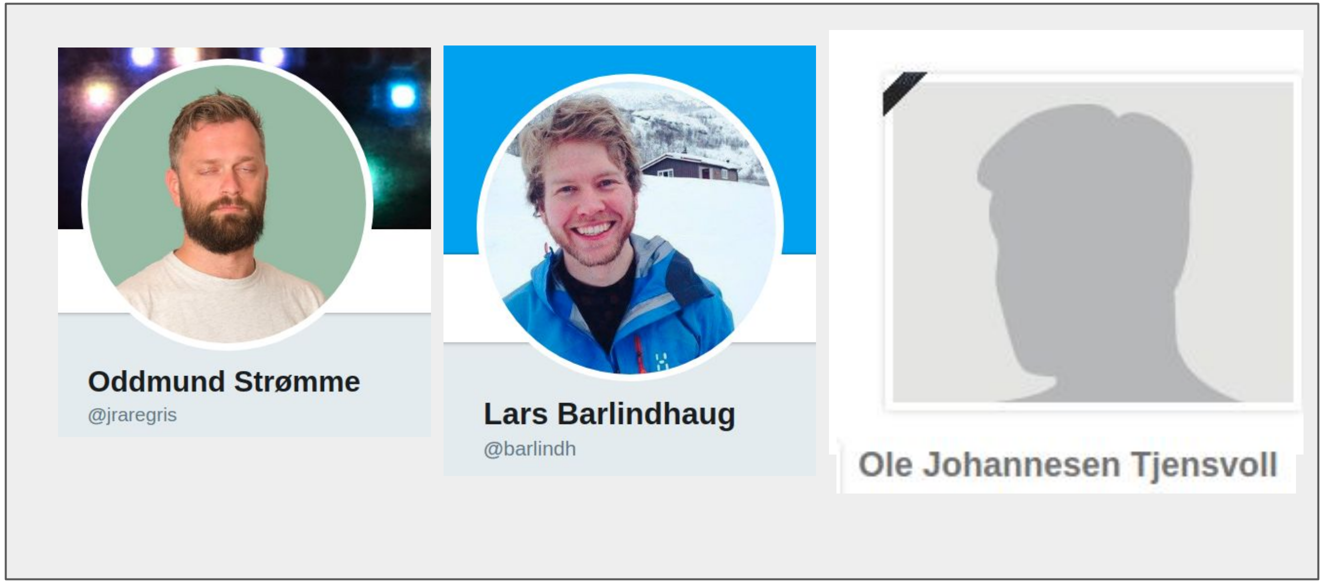


```
while(true);
 do
 git pull --rebase;
 git push;
 done;
```



```
test && git commit -am working
```

Test && Commit || Revert

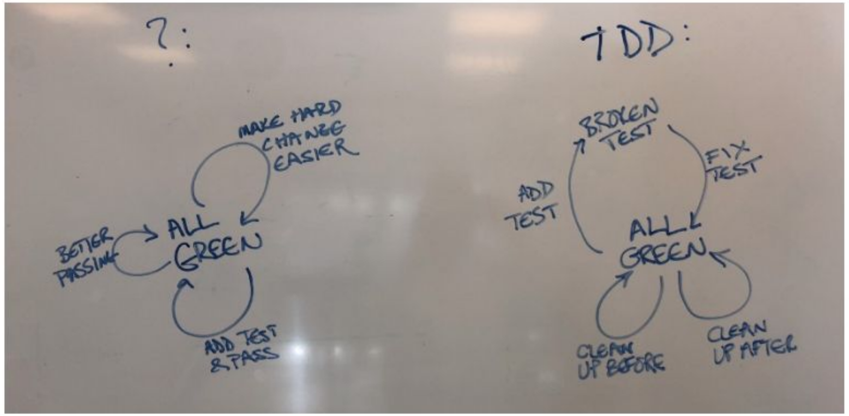


Symmetry

28.09.2018 TCR

test && commit || revert

Kent Beck
Sep 28, 2018 · 3 min read

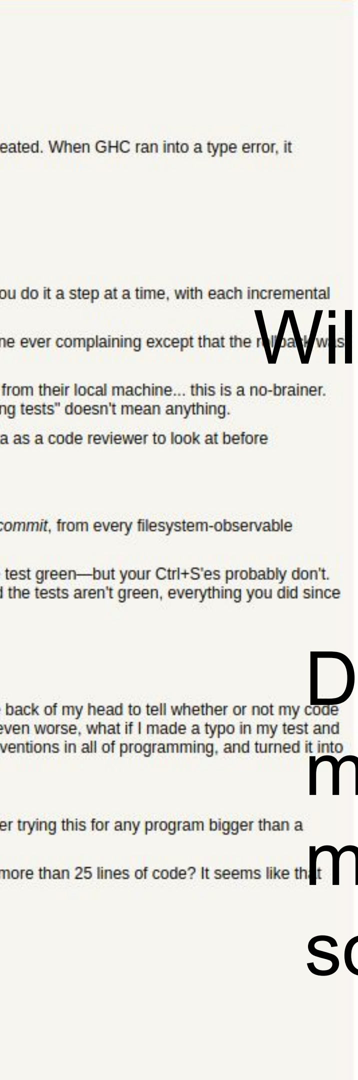


The new style versus test-driven development

As part of Limbo on the Cheap, we invented a new programming workflow. I introduced “test && commit”, where every time the tests run correctly the code is committed. Oddmund Strømmer, the first programmer I’ve found as obsessed with symmetry as I am, suggested that if the tests failed the code should be reverted. I hated the idea so I had to try it.

Top highlight

The full command then is “test && commit || revert”. If the tests fail, then the code goes back to the state where the tests last passed.


eated. When GHC ran into a type error, it

ou do it a step at a time, with each incremental

ne ever complaining except that the rollback was

from their local machine... this is a no-brainer.
ng tests" doesn't mean anything.

a as a code reviewer to look at before

commit, from every filesystem-observable

test green—but your Ctrl+S'es probably don't.
l the tests aren't green, everything you did since

back of my head to tell whether or not my code
even worse, what if I made a typo in my test and
ventions in all of programming, and turned it into

er trying this for any program bigger than a

more than 25 lines of code? It seems like that

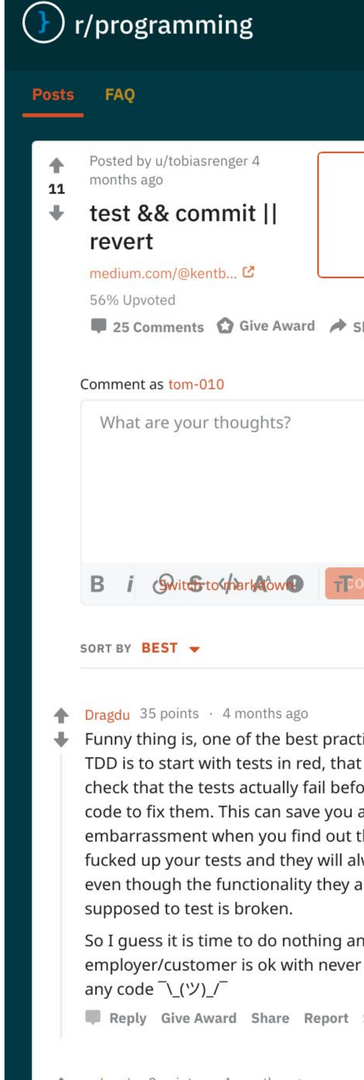
Will not Work!

Won't you get frustrated?

How could you make progress?

Don't you make mistakes sometimes?

What if you write a bunch of code and it just gets wiped out?


r/programming

Posts FAQ

11

Posted by u/tobiasrenger 4 months ago

test && commit || revert

medium.com/@kentb...

56% Upvoted

25 Comments Give Award

Comment as tom-010

What are your thoughts?

Switch to markdown

SORT BY BEST

Dragdu 35 points · 4 months ago

Funny thing is, one of the best practi
TDD is to start with tests in red, that
check that the tests actually fail befo
code to fix them. This can save you a
embarrassment when you find out t
fucked up your tests and they will alv
even though the functionality they a
supposed to test is broken.

So I guess it is time to do nothing an
employer/customer is ok with never
any code ¯_(ツ)_/¯

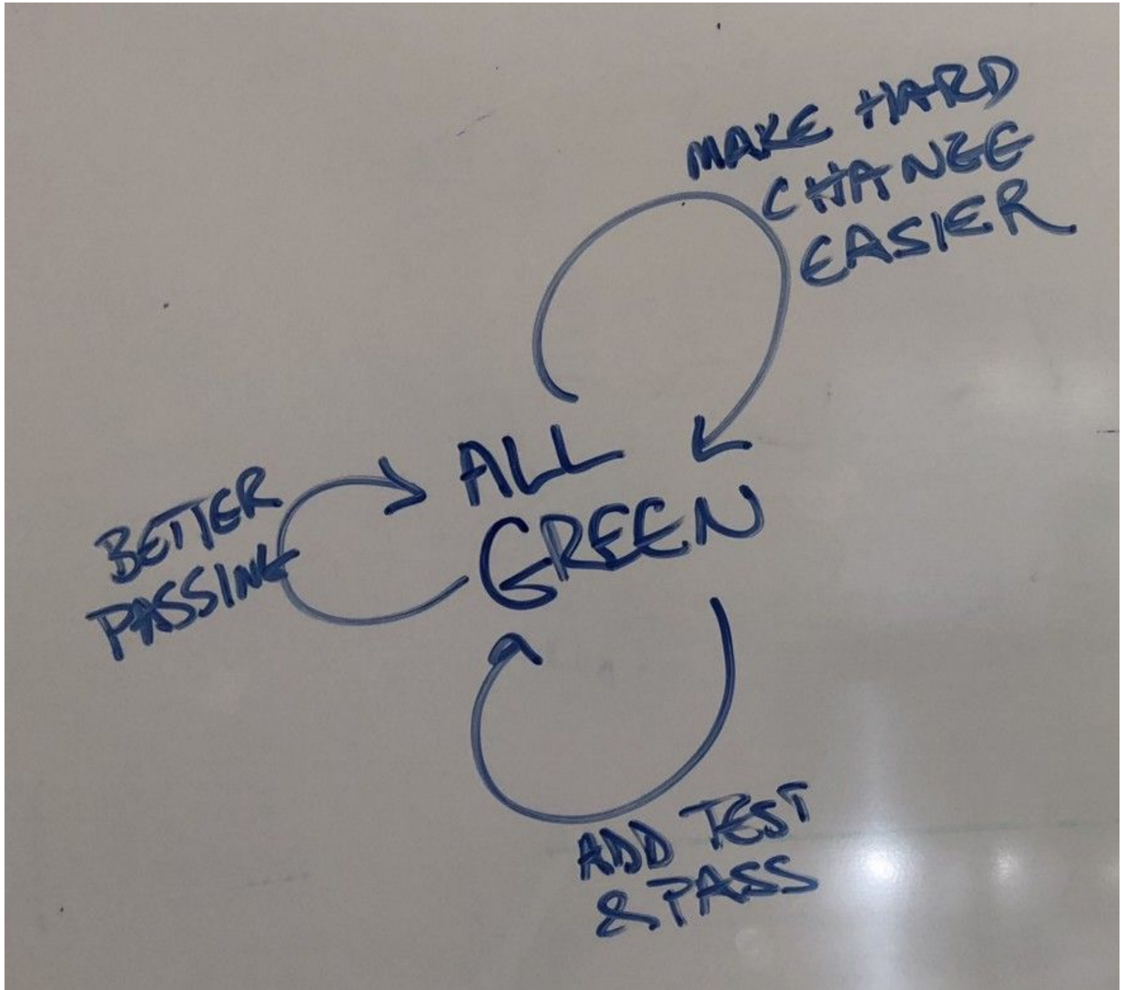
Reply Give Award Share Report

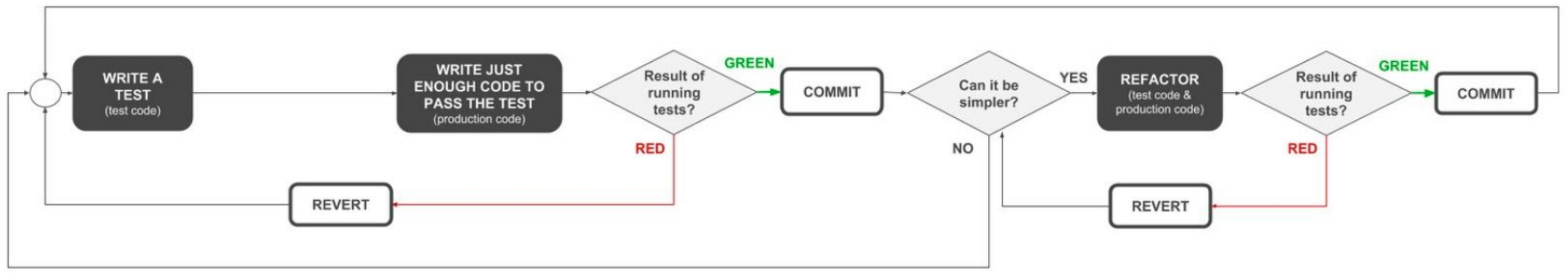
. I hated the idea so I had to try it.

(Ab-) using Git

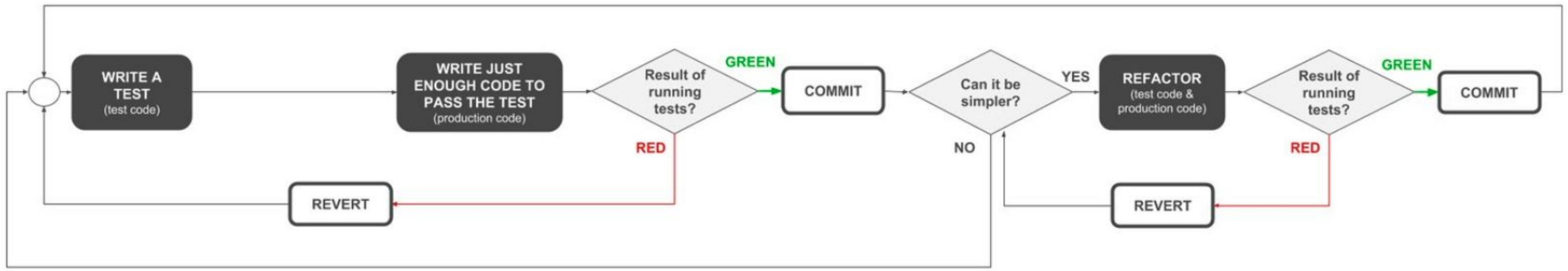
```
./test.sh && git commit -am working || git reset --hard
```

Always Green





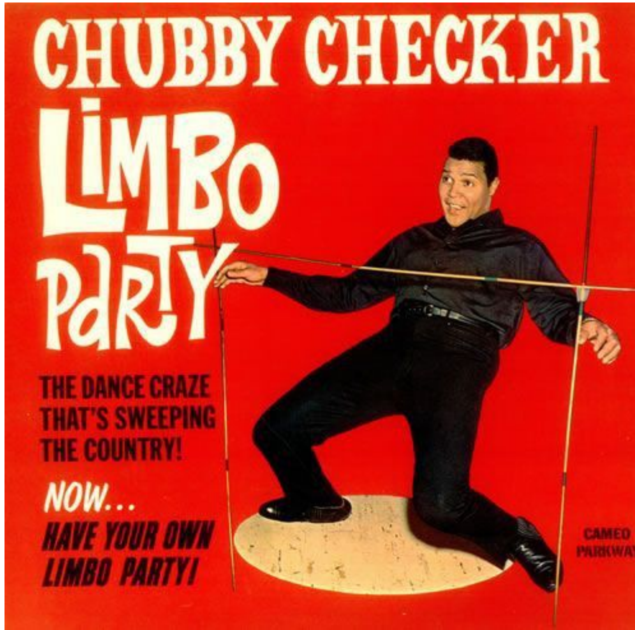
By Rachel M.Carmena



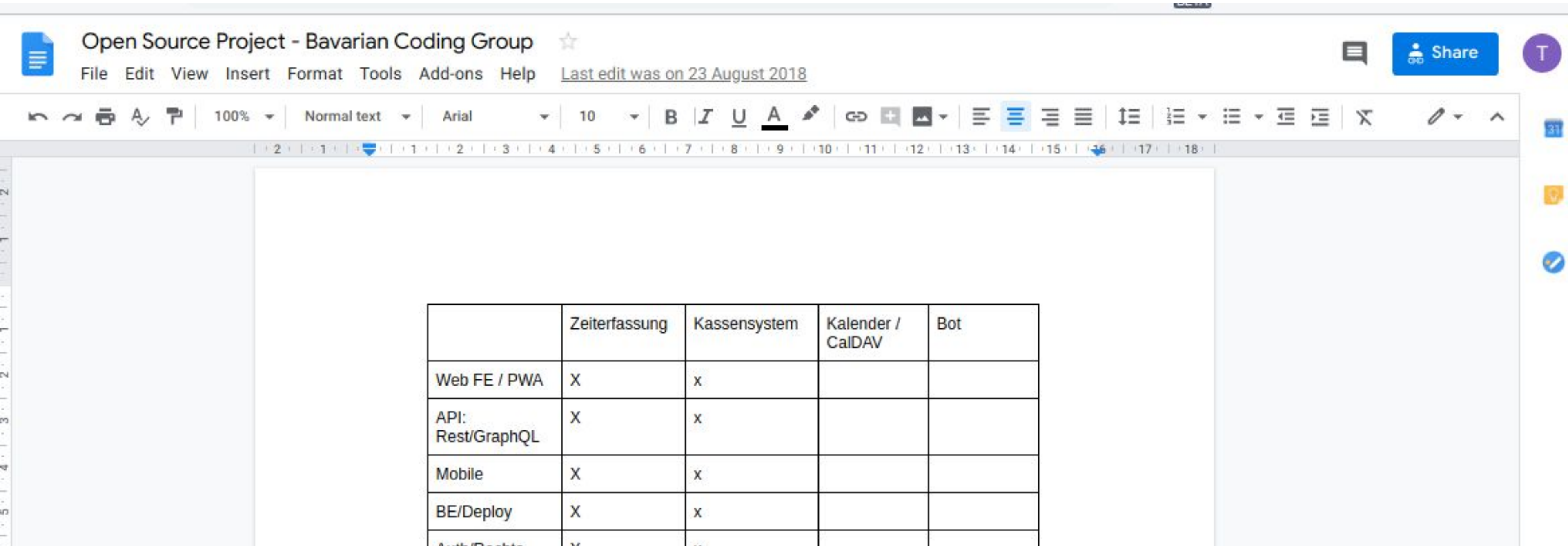
By Rachel M.Carmena

Conceptional View

How low can you go?

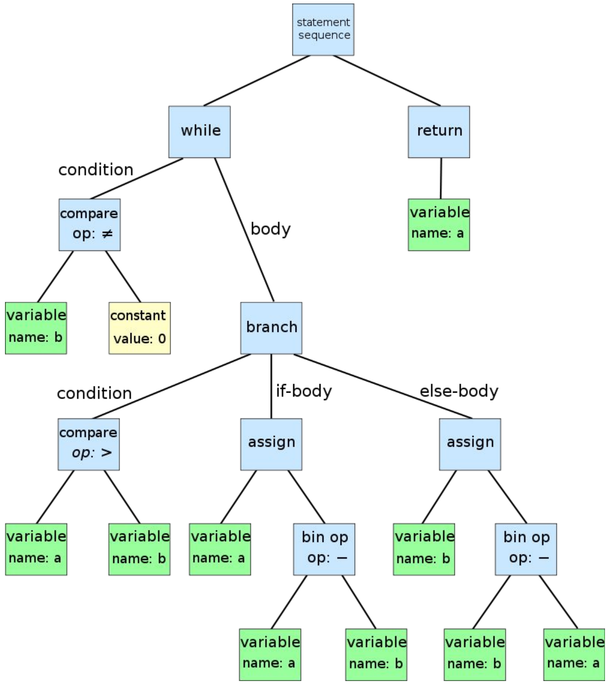

CHUBBY CHECKER
LIMBO PARTY
THE DANCE CRAZE THAT'S SWEEPING THE COUNTRY!
NOW... HAVE YOUR OWN LIMBO PARTY!
CAMEO PARKWAY

Ideal: Google Docs

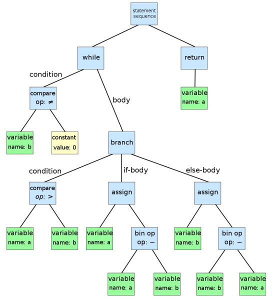

Open Source Project - Bavarian Coding Group
File Edit View Insert Format Tools Add-ons Help
Last edit was on 23 August 2018
Share
100%
Normal text
Arial
10

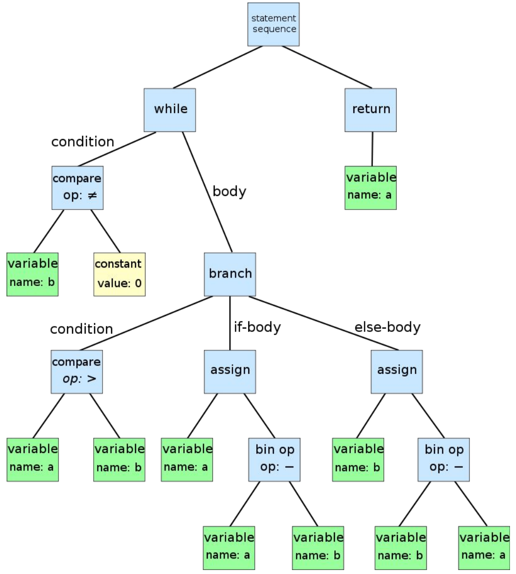
	Zeiterfassung	Kassensystem	Kalender / CalDAV	Bot
Web FE / PWA	X	x		
API: Rest/GraphQL	X	x		
Mobile	X	x		
BE/Deploy	X	x		

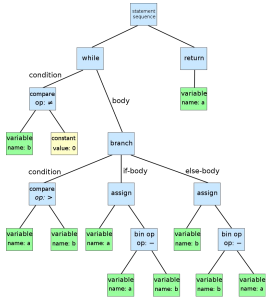
Source Code != Documents

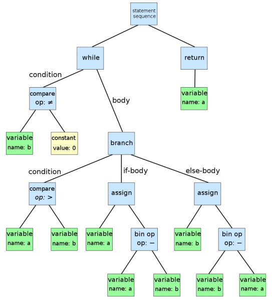


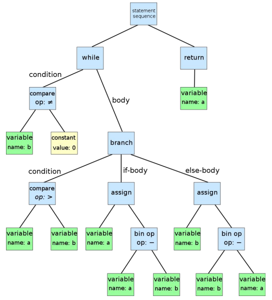
AST Transformations











A Git Prototype

```
test && commit
```

Limbo Principles

1. Everyone is working on (and production is executing) the same program, represented by a single abstract syntax tree.
2. No one is allowed to cause others (including users) problems.

From Limbo to a Git Workflow

```
while(true);
do
  git pull -- rebase;
  git push;
done;
```

To TCR

```
test && commit || revert
```

```
test && commit
test || revert
```

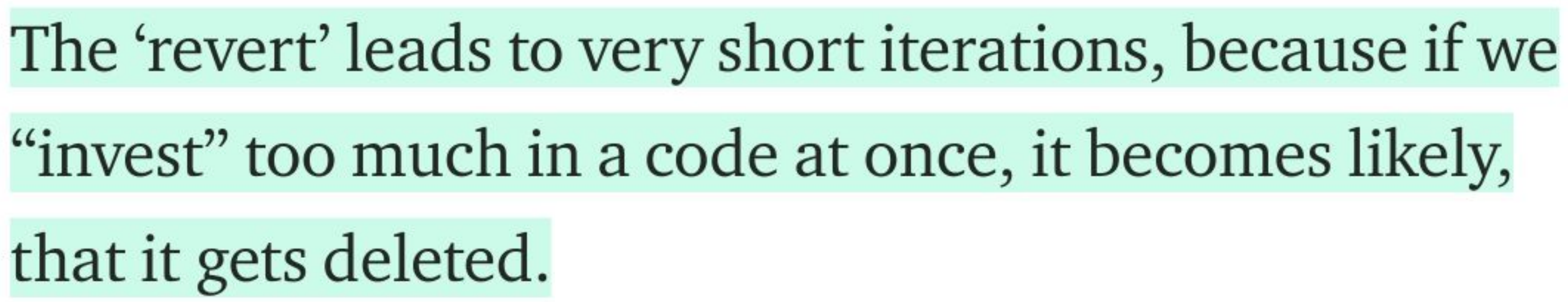
To TCR

Limbo
Collaboration

```
test && commit
test || revert
```

TDD
Workflow

Change the Act of Programming


The ‘revert’ leads to very short iterations, because if we “invest” too much in a code at once, it becomes likely, that it gets deleted.

TDD

1. We **begin** in Green (do: `test`)
2. We create a **failing Unit-Test** and are in red (do: `test`)
3. We **fix** the test to be Green again (do: `test`—as often as necessary to arrive in Green)
4. We **refactor**. We are in green and stay in green (do: `test`)

TDD

1. We **start** Green (do: `test && commit || reset`)
2. Write a **test** (do: `test && commit || reset src`)
3. **Fake** the implementation (do: `test && commit || reset src`)
4. **Refactor**, where you replace n Fakes with a real implementation (do: `test && commit || reset src`)

TCR Variants (test && commit || revert)

Thomas Deniffel
Nov 16, 2018 · 5 min read

The Original

The schema of the original version is ‘test && commit || revert.’ As an answer to the question regarding the particular commands, Kent suggested:

```
$ ./test && git commit -am working || git reset — hard
```

```
if(test().success)
    commit()
else
    revert()
```

BTCR

The first alternative solves the compilation issue.

```
$ ./buildIt && (./test && git commit -am working || git reset — hard)
```

```
if(build().failed)
    return

if(test().success)
    commit()
else
    revert()
```

The watch buddy

Alejandro Marcu pointed out, that the infinity-loop is a waste of resources. ‘The watch buddy’ brings an optimization as it introduces a file-system-watch (e.g. ionotify in Linux):

```
while true
do
    inotifywait -r -e modify .
    ./tcr
done

$ cat tcr
./buildIt && (./test && git commit -am working || git checkout HEAD —
src/main/)
```

The watch buddy

Alejandro Marcu pointed out, that the infinity-loop is a waste of resources. ‘The watch buddy’ brings an optimization as it introduces a file-system-watch (e.g. ionotify in Linux):

```
while true
do
    inotifywait -r -e modify .
    ./tcr
done

$ cat tcr
./buildIt && (./test && git commit -am working || git checkout HEAD —
src/main/)
```

```
while(true) {
    block_until_change_in_directory('src')
    tcr()
}

function tcr() {
    if(build().failed)
        return

if(test().success)
        commit()
    else
        revert()
}
```

The Original

BTCR

The Collaborator

The watch buddy

The Relaxed

The Gentle

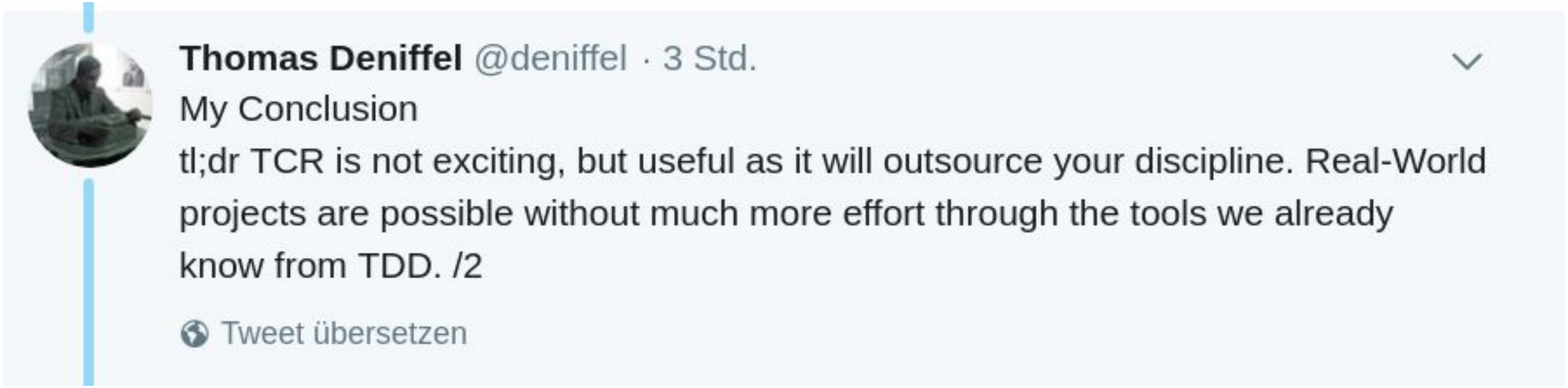
The Split

The Storyteller

Local Buddy, Remote Team

The buddy—Continous TCR

TCR in Practice



Thomas Deniffel @deniffel · 3 Std.

My Conclusion

tl;dr TCR is not exciting, but useful as it will outsource your discipline. Real-World projects are possible without much more effort through the tools we already know from TDD. /2

Tweet übersetzen

All resources on TCR annotated

Thomas Deniffel
Feb 21 · 2 min read

July 2018

@deniffel

Thomas Deniffel

@deniffel

We must know. We shall know. (Hilbert) | Programming Philosopher and Craftsman at Skytala GmbH