Limited WIP for developers
<table>
<thead>
<tr>
<th>Backlog</th>
<th>Design</th>
<th>Development</th>
<th>Testing</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backlog</td>
<td>Design</td>
<td>Development</td>
<td>Testing</td>
<td>Done</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Work-in-progress (WIP) limits:
- Backlog: 3
- Design: 5
- Development: 3
<table>
<thead>
<tr>
<th>Backlog</th>
<th>Design</th>
<th>Development</th>
<th>Testing</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Work-in-progress (WIP) limits

3  
5  
3
<table>
<thead>
<tr>
<th>Backlog</th>
<th>Design</th>
<th>Development</th>
<th>Testing</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Work-in-progress (WIP) limits

- Backlog: 3
- Design: 5
- Development: 3
<table>
<thead>
<tr>
<th>Backlog</th>
<th>Design</th>
<th>Development</th>
<th>Testing</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
</tr>
<tr>
<td></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
</tr>
<tr>
<td></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
</tr>
<tr>
<td></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
</tr>
<tr>
<td></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
<td><img src="note.png" alt="Note" /></td>
</tr>
</tbody>
</table>

Work-in-progress (WIP) limits:
- 3
- 5
- 3
THE GOAL
A PROCESS OF UNCONSCIOUS CONTROL
25th Anniversary Edition

Eliyahu M. Goldratt

Includes case study interviews

THE BEST-SELLING BUSINESS NOVEL THAT INTRODUCED THE THEOREY OF CONSTRAINTS
AND CHANGED HOW AMERICA DOES BUSINESS

OVER 4 MILLION COPIES SOLD!
THIRD REVISED EDITION

“Goal” readers are now doing the best work of their lives.”
Success Magazine

“A factory must be an unlikable setting for a novel, but the book has
been wildly effective...”
Tom Peters

Eli Goldratt has been described by Fortune as a “guru to industry” and by Business Week as a “genius”. His book, The Goal, is a gripping fast-paced business novel.
# Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Compatibility with IntelliJ IDEA Ultimate</th>
<th>Update Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.12</td>
<td>2020.1+</td>
<td>Feb 14, 2021</td>
</tr>
<tr>
<td>2020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.11</td>
<td>2020.1+</td>
<td>Jun 05, 2020</td>
</tr>
</tbody>
</table>
Other Settings > Limited WIP

Change size watchdog

- Enabled: checked
- Change size threshold: 120 lines
- Notification interval: never
- Show remaining changes in toolbar: checked
- No commits above threshold: unchecked
- Excluded files: *

Auto-revert

- Enabled: checked
- Timeout period: 5 minutes
- Notify on revert: checked
- Show time till revert: checked

TCR mode (test && commit || revert)

- Enabled: checked
- On passed test: commit
- Use commit message from: last commit
- Notify on revert: unchecked
- Don't revert tests: unchecked
- Don't revert files: *

Open readme on GitHub
FizzBuzzWoof
FizzBuzzWoof

- if divisible by 3, it’s “Fizz”
FizzBuzzWoof

- if divisible by 3, it’s “Fizz”
- if divisible by 5, it’s “Buzz”
FizzBuzzWoof

- if divisible by 3, it’s “Fizz”
- if divisible by 5, it’s “Buzz”
- if divisible by 7, it’s “Woof”
FizzBuzzWoof

- if divisible by 3, it’s “Fizz”
- if divisible by 5, it’s “Buzz”
- if divisible by 7, it’s “Woof”
- and combinations…
FizzBuzzWoof

- if divisible by 3, it’s “Fizz”
- if divisible by 5, it’s “Buzz”
- if divisible by 7, it’s “Woof”
- and combinations…
- otherwise, it’s the number
Agenda

1) TDD with auto-revert
2) Test & commit || revert
3) Takeaways & conclusion
Part 1
TDD with auto-revert
TDD with auto-revert
(and change size limit)
Mini-retro
Mini-retro
- stress from the timer
Mini-retro

- stress from the timer
- rushing to commit
Mini-retro
- stress from the timer
- rushing to commit
- smaller changes +
regular commits = 😎
Coderetreats are free day-long, intensive practice events, focusing on the fundamentals of software development and design. By providing developers the opportunity to take part in focused practice, away from the pressures of "getting things done", the coderetreat format has proven itself to be a highly effective means of learning and nurturing software development skills.

All over the world, people are coming together in small communities of practices to learn, explore and grow. Join us on the Global Day of Coderetreat or discover local communities around you!
Taking Baby Steps

Posted by Adi Bolboaca on Mar 11, 2013

I wrote a post on the history of Taking Baby Steps, you can read it here.

Now I would like to tell you more about the workshop and the technique itself. While coding, for me it is very important to be focused on one idea. Why? Because this is how programmer mistakes appear in the code; some people tell them bugs. The other thing I am interested in is to have an undo button for every change. This is why I want to commit every 1-2 minutes.

The rules are the following:

Steps

1. Setup source control repository.
2. Setup a timer for 2 minutes interval when you start.
3. Write exactly one test
   1. If the timer rings and the test is red then revert and start over.
   2. If the test is green before timer rings then commit.
Auto-revert timeout:
Auto-revert timeout:
- months / years
Auto-revert timeout:
- months / years
- days / weeks
Auto-revert timeout:
- months / years
- days / weeks
- minutes / hours
Auto-revert timeout:

- months / years
- days / weeks
- minutes / hours
- second(s)
Change size limit:
Change size limit:

- 10000000 lines
Change size limit:

- 10000000 lines
- 100000 lines
Change size limit:

- 10000000 lines
- 100000 lines
- 100 lines
Change size limit:

- 10000000 lines
- 10000 lines
- 100 lines
- 1 line
TDD is a WIP limiting technique 😐
Enabling constraints for the win!
Part 2
test &&
commit ||
revert
- can’t commit without running tests
- can’t commit without running tests
- if tests pass, commit changes
- can’t commit without running tests
  - if tests pass, commit changes
  - if tests fail, revert changes
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 Strømme, the first programmer I've found as
TDD:
- Broken Test
- Fix Test
- Add Test
- Clean up before
- Clean up after

ALL GREEN

Make hard change easier

Better passing

All Green

Add Test & pass
TCR
quirks
- incompatible with TDD
(can’t have failing tests)
- incompatible with TDD
  (can’t have failing tests)
- test stacktraces are not useful
- incompatible with TDD (can’t have failing tests)
- test stacktraces are not useful
- commit messages?
- incompatible with TDD (can’t have failing tests)
- test stacktraces are not useful
  - commit messages?
- commit or commit & push?
TCR Variants (test && commit || revert)

In his post, Kent Beck suggests ‘test && commit || revert.’ This a
import org.junit.jupiter.api.Test

class RegexMatcherTests {

    @Test fun `some examples`() {

        // "\"".matchesRegex("\"\") shouldEqual true
        "a".matchesRegex("a") shouldEqual true
        "\"\".matchesRegex("a") shouldEqual false

        // "a".matchesRegex("\."\") shouldEqual true
        // "\"\".matchesRegex("a") shouldEqual false

        // "a".matchesRegex("a\?\") shouldEqual true
        // "\"\".matchesRegex("a\?\") shouldEqual true
        // "b".matchesRegex("a\?\") shouldEqual false

    }
}

private fun String.matchesRegex(regex: String): Boolean {
    if (this.length != regex.length) return false
    return this.zip(regex).
        all { it.second == '.' || it.first == it.second }
}
Part 3
Try it yourself!
DIY
change size limit
via git hooks
DIY auto-revert
DIY TCR

gradle test &&
git commit -a ||
git reset --hard
# Limited WIP

**Dmitry Kandalov**

## Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Compatibility with</th>
<th>Update Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021</td>
<td>IntelliJ IDEA Ultimate</td>
<td></td>
</tr>
<tr>
<td>0.3.12</td>
<td>2020.1+</td>
<td>Feb 14, 2021</td>
</tr>
<tr>
<td>2020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3.11</td>
<td>2020.1+</td>
<td>Jun 05, 2020</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

Questions
What is the point of commit messages? 🤔
What is the point of commits? 🤔
Is commit/push the new “Save” button?
What about continuous updates?
Ultimate CI: continuous push/update on a single branch 🌟
Are our toolchains too slow for these workflows? Why?
Should WIP be applied end-to-end? (not just coding)
More feedback loops in IDE/editor?
Darklang?
Source code and links

https://github.com/dkandalov/limited-wip-for-devs
The End
@dmitrykandalov
github.com/dkandalov
youtube.com/dkandalov