Coding in the FLOW

Structuring your development environment to promote a state of flow

Caskey L. Dickson
SRE/SWE
@caskey@gmail.com

At the end are the lecture notes for the three slides we didn’t cover at oscon describing an application of flow.
tl;dw

1. Sprints/SCRUM/Rapid Development methodology (GT)
2. Revision control with local and feature branches (G)
3. One click build system (F)
4. Test driven development (GF)
5. Tiered testing (unit, smoke, integration) (GFS)
6. Top-down/bottom-up (GS)
7. Defect tracking (GS)
8. Dashboards (FT)
9. Blameless workplace (T)
10. Shared vision (GT)

G = Clear, attainable goals
F = Immediate and relevant feedback
S = Matched Skill and Challenge
T = Team dynamics

WARNING: Cargo Culting flow won’t really work.
Outline

1. tl;dw
2. Consciousness
3. Intentions
4. Limits of Consciousness
5. Attention
6. Attributes of Flow
7. Prerequisites of Flow
8. Structuring for Flow
[citation needed]
Flow: Definition

In positive psychology, **flow**, also known as **zone**, is the mental state of operation in which a person performing an activity is fully immersed in a feeling of energized focus, full involvement, and enjoyment in the process of the activity. In essence, flow is characterized by complete absorption in what one does.

– Mihaly Csikszentmihalyi
Consciousness

A human being should be able to change a diaper, plan an invasion, butcher a hog, conn a ship, design a building, write a sonnet, balance accounts, build a wall, set a bone, comfort the dying, take orders, give orders, cooperate, act alone, solve equations, analyze a new problem, pitch manure, program a computer, cook a tasty meal, fight efficiently, die gallantly. Specialization is for insects.

—Robert A. Heinlein
Intentions

- Sensations and stimuli
- Processed by the consciousness
- Turned into potential actions
- Exogenic
- Endogenic
- Prioritized (continuously)
- Selected for action
Limits of Consciousness

- Maximum of ~7 bits of concurrent information
- ‘bit’ varies by experience and subject matter
- Rate of recognition and change of bits 55ms
- Brain baud rate is therefore 126 bps
- Human speech recognition–40bps
- In a lifetime, 280Gbits of data
- Except
  - sleeping 20-35%
  - eating 8%
  - washing, dressing, toileting 8%
Attention

- Generally your consciousness chooses what to ingest
- What you ingest is merged with your intentions
- Emotional response occurs when those conflict
- External events can trigger natural attention sinks
- Every moment gone by is a spent opportunity to maximize or squander your 126 bits of focus

“Pay Attention” means exactly that.
The Self

- Consciousness exists
- Stimulus drives intentions
- Intentions are ranked by the consciousness
- Consciousness has finite capacity (126 bits/s of input)
- Our attention dictates inputs per the consciousness
- Therefore our consciousness dictates our mental state
High Flow Activities

- Music
- Rock Climbing
- Dancing
- Sailing
- Chess
- Sports
- Reading

- We hope: Coding ( ← You are here. )
Most Common Attributes of High Flow Activities

- Rules
- Goals
- Feedback
- Control
- Concentration
- Separation from Reality
RULES AND REGULATIONS

RULE V

It is illegal to macaroni of the macaroni machine to provide macaroni, and may not be cut under the machine. The macaroni is to avoid all losses, and the man who will never be able to machine must observe the following rules:

1. The macaroni must be pinched hand full into the machine which is the correct way of feeding the macaroni. Also the macaroni must be pinched in at a uniform speed, and in no case pile them upon the feeder.

2. It is demanded by the Government that the full wheel and eye shall be the floured first. Spreading will be threshold at times of threshold sale.

3. In regard to time for a day's work, we would recommend that the Government order be saved to have all the grain possible, as it is advantageous that the order of the day, owing to the morning's daily and the difficulty of doing good work in the early morning, we would recommend that the hours of quitting shall not be before 3 p.m., new time.

RULE VI

It shall be the duty of the machine man to avoid all flaws for the following reasons:

A. Throwing grain when it is tough (slippery and unripe)

B. Loss from starting to handle wagons.

C. Causing to stop the threshing cylinder up to speed, and to adjust of bowen, etc., shall and last death.

D. Considerations in feeding handles to loose grain into the machine.

E. Considerations in allowing grain to back on the ground around and under the macaroni and hazards, up to time of operation.

F. Improper adjustment of canvas and other parts of machine.

RULE VII. nuturing the tread. It shall be the duty of the farmer to see that all wagons handled are tight. Also the case to have the corn down from a moving and all the macaroni will be cleaned carefully. It is also recommended that it is impossible to make the wheel fields before the fields of the wheel threshing. It is ruled often, and threshold in the return of the macaroni for the main macaroni, also is one that no grain is left by the man who pitches on the checks: "Always remove any after each check and pick up all handles and dropped from wagons by the规程s.

RULES for kids using BATHROOM

1. Open Shower Curtain, Check for Monsters
2. Pee/Do your Business
3. Flush
4. Check For Monsters again, CLOSE Curtain!
5. Wash Hands, then Run

A. F. Paden
F. H. Faulkner
W. H. Machin

Treshing Committee of the U. S. Food Administr for Knox Co.
HFA Attributes: Goals
HFA Attributes: Feedback
HFA Attributes: Control
HFA Attributes: Concentration
HFA Attributes: Separation from Reality
Three Prerequisites of Flow

- Clear Goals
- Immediate Feedback
- Matched Skill and Challenge
Clear Goals

You have to know where you are going
Knowing “Why?” helps too
Journey of 1000 miles
Immediate Feedback

Emphasis on immediate
Matched Skill and Challenge

- Low Skill, Low Challenge: Boredom
- Low Skill, High Challenge: Frustration
- High Skill, Low Challenge: Flow Zone
- High Skill, High Challenge: Flow Zone

Caskey L. Dickson
@caskey@gmail.com
OSCON2015: Coding in the FLOW
Clear Goals

- Planned work, baby steps
  - SCRUM
  - Rapid Development
  - Agile
- Product Vision/Leadership
- Local Branches–Commit early and often (yay git!)
- Top-Down/Bottom-Up
- Test Driven Development
- Tiered Testing
  - Unit
  - Smoke
  - Integration
- Defect Tracking
Immediate Feedback

- One click build system (F)
- Test driven development (GF)
- Tiered testing (unit, smoke, integration) (GFS)
- Dashboards (FT)
- Smart Tools (GF)
Matched Skill and Challenge

- Sprints/SCRUM/Rapid Development methodology (GT)
- Test driven development (GF)
- Tiered testing (unit, smoke, integration) (GFS)
- Top-down/bottom-up (GS)
- Defect tracking (GS)
- Blameless workplace (T)
- Codecraft
Summary

Minimize **cognitive load** (126 bits per second)
Establish **Clear Goals** (final and intermediate)
Ensure **Immediate Feedback** (one click build+test, tools)
Match the **Skill to the Challenge** (increase if need be)
Questions, Comments, Complaints, Hate mail:

caskey@gmail.com
twitter.com/caskey
plus.google.com/+CaskeyDickson
tl;dw

1. Sprints/SCRUM/Rapid Development methodology (GT)
2. Revision control with local and feature branches (G)
3. One click build system (F)
4. Test driven development (GF)
5. Tiered testing (unit, smoke, integration) (GFS)
6. Top-down/bottom-up (GS)
7. Defect tracking (GS)
8. Dashboards (FT)
9. Blameless workplace (T)
10. Shared vision (GT)

G = Clear, attainable goals
F = Immediate and relevant feedback
S = Matched Skill and Challenge
T = Team dynamics

WARNING: Cargo Culting flow won’t really work.
Extras
Lecture notes for slide 23

There are lots of things that help us with our planning, but even detailed methods like daily standups probably are not narrow enough to be flow inducing.

- **fixing a single bug in your system.**
  - sounds fine grained task
  - flow is about short term tactical gains.
  - Measured in hours, not days/weeks
- fixing the bug is the ultimate goal, but we need to measure progress!
- must, necessarily find finer steps, 2, 3, 10.
  1) In my daily standup with my team, pick bug 339 as my main task for today, in concert with the overall team goal: stabilizing the next feature release
  2) Put on my headphones
    - choose music that has a white noise effect dull my auditory senses
    - enables me to focus
    - avoid getting pulled into the discussions
    - Separation from reality
  3) one-click build and run environment for this particular work session
    - bookkeeping of creating a new git branch
    - pulled from the appropriate code line or feature branch
    - named for the task at hand with cross references into our bug tracking system
    - resume flow if you go to lunch or get switched off to deal with something urgent
    - enables rapid resumption with minimal cognitive
  4) reproducible failing test case (TDD)
    - unambiguously feedback on progress
    - Usually several steps (nested processes)
      - failure isn’t easily testable
      - refactoring to surface the appropriate components.
  5) Critical at each stage: local commit to my working branch
    - ‘refactored class X to make initial state externalizable’
    - ‘fixed the breakage I induced in the integration test when I messed up refactoring class X’
    - ‘added failing unit test for bug 339’
  6) Now that I have an environment where I can hit F7 or whatever the appropriate key is, to begin actually solving the problem at hand.
  7) work until the failing test case passes.
  8) work until smoke tests pass
  9) work until integration tests pass
  10) Finally, I do the **merge** with the mainline branch after resolving all those wonderful merge conflicts.
  11) Kick myself for not doing a merge after I refactored class X and before I started the bugfix

- **Much is existing best practices**
  - Applied with an eye toward satisfying the needs of flow
- **Even I skip this workflow**
  - Often regret it
  - annoyed at my task
  - 4 hours in
  - opened the code base
  - started making changes
  - discovered the class X refactoring
  - started doing that on top of my partial bugfix
  - broke the build horribly
  - can’t unwind my bug fix code from my refactoring code
  - wish I had an intermediate checkpoint even an hour or two old
  - haven’t passed a test in three hours, haven’t successfully built in two
  - burn it down, start again
- **Also, Checklist Manifesto, read it.**
  - Discipline is hard, humans are bad at it
In that discussion about fixing a bug, when I did it the right way, I set up a system where I would get immediate feedback as to whether or not the problem had been fixed or not, including building parts of the system I would need in the refactoring.

Each step of the way all I needed to do was hit F7 (or whatever your dev environment equivalent is) and it would kick off a build + run of the relevant unit tests showing a big green or big red box. Bam, immediate feedback. Unit tests, not integration tests because they run faster (a few seconds at most).

- Even better, for some languages my IDE catches typos for me automatically and underlines them
  - Immediate feedback.
- For other languages, it tells me when I’ve forgotten to initialize a variable
  - Immediate feedback.
- For other languages, my editor automatically navigates me to the line and character of compiler errors while highlighting the error message in a contrasting color so I can continue working
  - Immediate feedback.
- Sadly for others, I have no type information or even required variable declarations and so I have no idea if my code really works until a full integration test. Very slow feedback. Given I’m very weak at writing code in this language, I often regret and put off projects that require it because I subjectively hate the lack of control I feel.

The polar opposite of this kind of immediate feedback is card based batch computing where your feedback loop is handing a program deck over to an operator, waiting an hour (or a week) and getting back your output deck or just a printout of the core dump.
In this area, it mostly comes down to choosing what to work on and, more importantly, choosing HOW to work on it.

- Learning a new system
  - Get it to compile
  - Get it to not crash
  - Get it to pass a trivial unit test
- Multi year wrestling match with Go
- Other languages where I have a couple decades experience
  - Mental effort goes to meta activities
    - clean code structure
    - readability
    - maintainability
    - testability
  - These reduce cognitive load of ongoing work
- Go look at your really old code
  - Try not to cry; cry a lot
  - By the standards at the time, you were doing well
- Enlightened teams
  - actively structure tasks, avoid all urgent work going to fastest/most experienced
  - manage work to promote zone and growth
- Individual contributor
  - Examine your tasks
  - Find ways to adjust the skill level up for the mundane/boring work
  - Focus on your craft, the skill
  - Drudgery always exists (Boredom zone)
    - At the meta, don’t do the drudgery, eliminate it entirely, automate it, write a tool, eliminate the need
  - Adjust your subjective experience and intent
- Proactively decide what kind of work you have and what sort of work you will do. Avoid boredom at all costs.