Why is it so hard to play like a KISS

By Pierluigi Fornoni

Table of Contents

The beginning..

Louis Daniel Armstrong (August 4, 1901 – July 6, 1971)

Louis Armstrong’s improvisations permanently altered the landscape of jazz by making the improvising soloist the focal point of the performance.

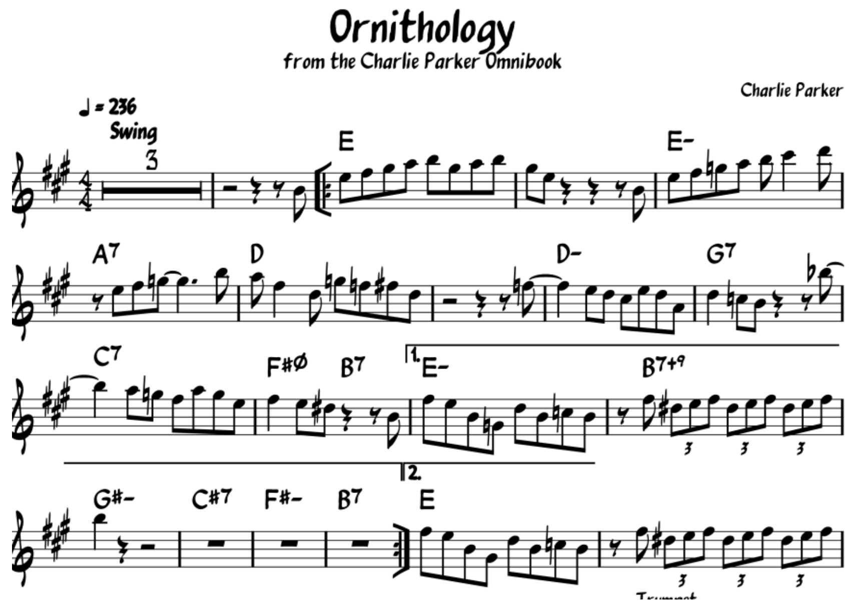
Evolution: from Classic Jazz to Bebop

- 1920 Early jazz, New Orleans (Armstrong)

- 1930 Swing Big Band (Ellington, Count Basie,…)

- 1940 Bebop

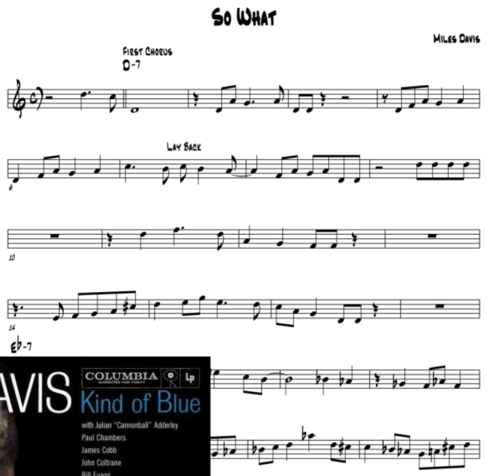
Increasingly complex chord changes over which soloists improvised

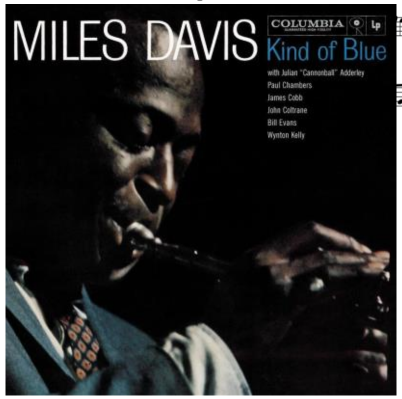


1959 The «Modal» Revolution

Miles Davis (1926-1991) recorded ***Kind of Blue***

- Easier to understand for the listener
- Comfortable to "extend" for the improviser





Where most forms of jazz had built increasingly complex chord changes over which soloists improvised, Davis shifted to using modes as the basis for song structure. Modal composition allowed Davis to slow down and **simplify** the structure of the piece.

What is KISS

KISS, an acronym for **keep it simple, stupid**, states that most systems work best if they are kept simple rather than made complicated;
therefore, simplicity should be a key goal in design, and unnecessary complexity should be avoided.

— wikipedia

Perfection is achieved, not when there is nothing more to add,
but when there is nothing left to take away.

— Antoine de Saint-Exupery

Why KISS ?

Kelly Johnson (1910-1990), was the lead engineer at the Lockheed Skunk Works

He told the designers that whatever they made had to be something that could be repaired by a man in a field with some basic mechanic's training and simple tools.

If their products weren't simple and easy to understand – they would quickly become obsolete in combat conditions and thus worthless.

Why is it so hard to be simple?

The theory is good, but the practice is a different league.

If we know and agree kiss principles, why we still lack in implementing them?

Reason 1: We are humans after all

We don’t realize that an easier solution exists. Sometimes for lack in business knowledge or misunderstandings, sometimes for technical part.

Solution:

- ➢ More engagement between business actors and developers
- ➢ Knowledge of principal code smells and refactoring patterns

"THIS IS NOT WHAT I MEANT WHEN I TOLD YOU TO GET IN TOUCH WITH YOUR FEMININE SIDE!!"

Reason 2: Speed

Change a line of code is faster and easier and less error prone than modifying class relations.

If something turns out to be «smelling», we will fix it next iteration, not thinking that in next iteration we we'll have same pressure and same complexity as in this one.

Solution:

- Test coverage + continuous refactoring
- Fight against temptation

I GENERALLY AVOID TEMPTATION UNLESS I CAN'T RESIST IT.

Reason 3: Influence

Spending time in refactoring + better process knowledge is sometime hard and slow.
It happens that management underestimate it and put some kind of pressure.

It’s just another «if», why you made so complicated?

Solution

- Resist (politely)
- Better management engagement in process
- Make small changes fast by applying TDD and KISS in previous iterations

Reason 4: YAGNI (Thinking Ahead)

Let's say you need to create a function that calculates the sum of 2 and 3. How would you implement it?

Amateur

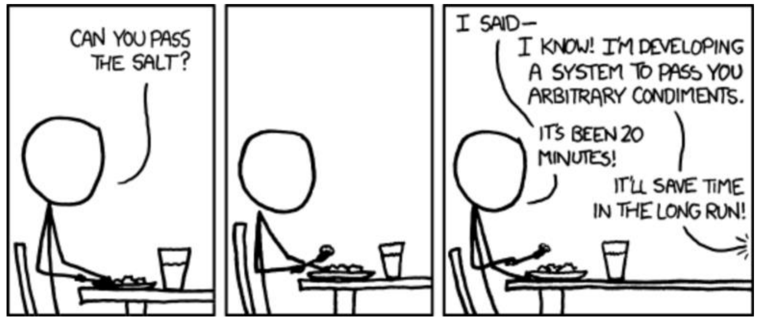
```
public int sum(int x, int y) {
  return x + y;
}
```

Pro

```
public int sumTwoAndThree() {
    return 2 + 3;
}
```

Solution

- Always implement things when you actually need them, never when you just foresee that you need them


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

Reason 5: Abuse of Patterns

For human reasons (and Dunning Kruger low), sometimes we abuse of what we think to know.

Solution

- Always implement things when you actually need them as simply as possible
- Never introduce an interface or abstract class for only one implementation. Create an abstraction only when you actually need it.

Reason 6: No need to be KISS

KISS can be in some conditions ignored (really!)
In particular if the code you produce will not be maintained (POC, …)
But be honest… it never happens.

Solution

- ➢ Be realistic on expected life duration of code you produce. There are little chances that all code you did will be re-developed from scratch after a demo

Conclusion

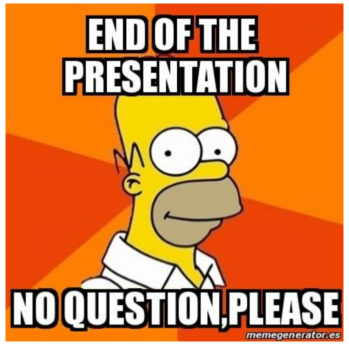
If you need to choose between two solutions, **choose the simplest one**.

Constantly work on simplifying your code base.

There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.

References:

❖ The Jazz Theory Book (Mark Levine 1995)
❖ Kinf Of Blue (Miles Davis, Columbia 1959)
❖ A Detailed Explanation of The KISS Principle in Software
❖ KISS (Keep it Simple, Stupid) - A Design Principle



Thank you

Pierluigi Fornoni, software engineer-eoc
pierluigi.fornoni@eoc.ch