

Real OOP

Raphael Diener 29.04.24

I originally wanted to talk to you about Object oriented programming.

I had all the slides prepared with jokes and infographics and mnemonics, and stuff.

But I was on a stroll in the woods, when I noticed the profound uselessness of the talk.

~~Real OOP~~

So I'm doing none of that today.

Instead I'm gonna talk to you about why you can't learn anything from a 15 minute presentation, using a 15 minute presentation.

Leaning

Neurology tells us that we learn by building mental models in our head.

There are a few underlying ways we might build mental models, but the most primitive and effective one is:

Neurons that fire together, bind together.

And this will most likely not be the case in a 15 minute presentation

Let's take the example of making a mental map.

The first day at UCB you were probably a bit disorientated.

You walked around until you found your lecture room.

The second time was faster and the path got strengthen.

Then after you have walked the path multiple times, it is strongly physically wired inside your brain.

sources:

Every second talk at: <https://www.youtube.com/@MITCBMM>

If you are looking for one to start with, try those two:

- [Hippocampal memory, cognition, and the role of sleep \(part 1\)](#)
- [Hippocampal memory, cognition, and the role of sleep \(part 2\)](#)

Also:

- [Jeff Hawkins - A Thousand Brains: A New Theory of Intelligence](#)

Repetition

"So repetition is required to strengthen connections between neurons.

We get it"

Yes, but this not something I can do.

If I were to constantly repeat myself, I'd probably only get you bored and a bad grade too.

And if I don't you won't take anything away.

You won't be learning anything from a 15 minute presentation.

Failure

But repetition alone is not enough.

What repetition can give you is nothing more than knowledge.

Repetition can't give you understanding.

Understanding comes from failure.

Failing is the best thing that can happen in your learning.

Some prominent voices also argue that it's the only way you can ever learn anything.

Back in our internal map example:

If you took the wrong turn once,

you will remember that spot the next day with:

"Oh, that's the spot I took the wrong turn".

The memory associated will be stronger and more salient.

But this is not something you can do inside a 15 minute presentation.

There is only one fail state for you and that is not listening.

And even if I bore you so much that you start sleeping,

this isn't a real fail state, is it?

The only thing you do during a presentation is listen.

And you've mastered that skill hopefully in the Kindergarten.

You won't be learning something *new* inside a 15 minute presentation.

Source:

Again, every psychologist I know would agree.

So, I'm citing [Dr.K.](#)

as well as the basis for the psychology lectures I had at Saarland University:

- Eysenck, M. & Keane, M. T. (2020). Cognitive psychology. A student's handbook (7th ed.). London: Psychology Press.
- Buchner & Brandt (2017). Gedächtniskonzeptionen und Wissensrepräsentationen. In J. Müsseler (Hrsg.), Allgemeine Psychologie, 3. Aufl.). Heidelberg: Spektrum.

- Wentura, D. & Frings, C. (2013). Kognitive Psychologie. Wiesbaden: Springer VS.

Anki

And we can see how integral failure is to learning when examining Anki as a case study.

You make flashcards to help you remember knowledge.

Not even understanding.

And yet, for effective memorization of knowledge, the cornerstone is not really repetition, but repeated failure.

Flashcards are made for you to fail.

And we know that flashcards are (one of) the most effective way to gain knowledge, since it is used by medical students all around the globe for exactly this reason.

So if even repetition fails in comparison to failing, it becomes even more obvious, that you won't learn something from a 15 minute presentation.

So, this was pretty much all I have to say on this matter.

But since we have so much time left,

how about we spend the rest of the time talking about the original topic.

OOP

Do you think Java is an object oriented language.

C++? C-sharp?

Huh, that's curious.

Those languages are class based, not object based.

Don't you find it curious why many people say that they are object oriented?

sources:

Yes, it really is commonly spread misinformation.

For my original presentation I went ahead and took a look at the first few distinct google search results.

They stated the wrongly what OOP was meant to be

and then following that quotes languages that fit their made up feature list.

Component	Times mentioned
encapsulation	16
inheritance	15
Objects with data and behavior	14
polymorphism	13
abstraction	12

Component	Times mentioned
reusability	7
class	6
modularity	6
extensibility	5
security	4
interface	4
design patterns	2
efficiency	2
coupling	1
cohesion	1
messages	1
persistence	1
scalability	1

In the table I've listed the components according to the following sites.

- techtarget.com
- spiceworks.com
- educative.io
- geeksforgeeks.org
- developer.mozilla.org
- wikipedia.org
- indeed.com
- freecodecamp.org
- w3schools.com
- realpython.com
- codeinstitute.net
- techtarget.com
- learn.microsoft.com
- coursera.org
- freecodecamp.org
- adv-r.hadley.nz
- sciencedirect.com
- w3schools.com
- mathworks.com

The definition by assoziation is meaningless but lets pick polymorphism as an example, mostly because I like that word.

There are multiple different kinds of polymorphism:

- Subtype Polymorphism
- Parametric Polymorphism
- Ad Hoc Polymorphism
- Row Polymorphism
- (Rank Polymorphism)
- (Coercion Polymorphism)
- ...

[Wikipedia - Polymorphism](#)

Lets make the assoziation as broad as possible and say that as long as it supports any kind of polimorphism,

it is considered an OOP-language.

Even then I can make a language that adheres to a better definition of OOP by the creator ([wiki.c2](#))

and doesn't implement any kind of polimorphism.

I'm actually doing that right now to test out different schedulers.

<https://github.com/AddGaming/C-OO-Interpreter>

Repetition legitimizes

I'd recon it has something to do with this. Doesn't it?

You've heard this factoid repeated so often, that you didn't even question it.

So the question becomes not only, why you repeated that piece of misinformation, but why the people who told you that, also repeated that piece of misinformation them self.

So how does information exchange in our industry primarily happen?

Sure, some read and write blog posts. Sometimes people also write books.

But I'd say that a non insignificant portion comes from here.

Conferences.

And there we have the problem.

You can't learn something from a 1 hour presentation.

When OOP started to emerge in the 80s,

there were only a hand full of small talk programmers and a couple erlang programmers, who actually understood what object orientation meant.

sources:

Here are some links describing how Erlang and Small-Talk work.

The only two real OOP languages I know of

[Systems that run forever self-heal and scale - Joe Armstrong](#)

[Lukas Larsson - Understanding the Erlang Scheduler](#)

[Seminar with Alan Kay on Object Oriented Programming \(VPRI 0246\)](#)

And to illustrate the problems of classical compute paradigms I recommend this talk:

[Reactive Systems • Dave Farley • GOTO 2019](#)

It shows how the message passing pattern of thought helps structure computation.

Alan Kay

Then they went to a conference or gave seminars on that topic and as expected, people didn't learned anything from listening to them.

People came from a mostly Algol, Fortran and Cobol background.

And when they saw all those foreign things inside OOP, they latched onto what they already knew.

And that was structs.

When people came back to the office, all they remembered was "*something something classes*".

And here I was foolishly thinking, that I could do better.

Better than over 40 years of knowledge transmission did before me.

If you want people to learn something, they have to have an foundation already, on which your new thing fits.

If you have something new which they can't intuit, they will not take it away.

And like the programmers in the 80s, your foundation is in C.

You would have learned nothing from a 15 minute presentation.

Japanese

Indulge me for a second talking about my study of the Japanese language.

For the N4 test I learned a total of 1500 words.

I knew them all for the exam.

If you were to ask me now how many of those I'd remember,

I would be impressed with myself if I were able to remember one fifth.

But if we talk about programming,

I'm pretty sure I could tell you about **All** of my failures.

You could throw me into a new language I've never seen before and I would be productive at the end of the same day.

Why is that?

Quality of Learning

The answer is, that not all learning is created equally.

I already mentioned it earlier, but learning through failure is considered better than learning through repetition.

But why?

The length of retention is directly correlated to the amount of connected information.

When you build understanding you do that explorative.

You have a central question or topic and then go explore in widely different directions.

The end result is a highly interconnected web of information,

as well as meta information about how you acquired that knowledge in the first place.

You can't learn anything from a 15 minute presentation.

Wait a minute.

I'm pretty sure, we all know a place where we spend a lot of time getting talked to a lot, under the pretense that we should be able to learn something.

sources:

The keyword that's related to this one is "Verarbeitungstiefe" in the German literature.

It's also discussed in the books I already linked above:

- Eysenck, M. & Keane, M. T. (2020). Cognitive psychology. A student's handbook (7th ed.). London: Psychology Press.
- Buchner & Brandt (2017). Gedächtniskonzeptionen und Wissensrepräsentationen. In J. Müsseler (Hrsg.), Allgemeine Psychologie, 3. Aufl.). Heidelberg: Spektrum.
- Wentura, D. & Frings, C. (2013). Kognitive Psychologie. Wiesbaden: Springer VS.

College

note:

But if we entertain the facts I mentioned previously seriously, then we can not help but end up with the realization, that teaching is one of the biggest obstructions to learning there is.

While you are getting taught you can't explore different paths.

You can't answer burning questions required for your understanding.

If you have sat in school for long enough you even learn to stop asking those questions in the first place.

You have regressed the ability to understand where your own knowledge lacks, and how to ask the questions to fill in those gaps.

That sheds an uncomfortable light on what you guys have been doing for the past few years, doesn't it?

Because you will not learn something from a 1 and a half hour lecture.

I mean, I've learned a lot from conference talks, lectures and video recordings, but none of that from the first time listening.

When I resonate with something or think there was something of meaning inside that piece of content, I read, listen, watch it again and again and again hoping that I can retain the knowledge for long enough, until I'm able to understand it.

But that's not something you can do in the lectures I'm taking this semester, because no lecturer records any of their lectures!

And you will not be able to learn something by listening to their 1 and a half hour lecture for the first time.

sources:

Since this section was about schools and learning, I'd be remissed not to mention the source the [Human Restoration Project](#).

I sadly can't quote one or two resources for this claim,

since it will play out in many different ways depending on where you look.

Even tho the general picture will remain, I personally know of no meta study to link.

Hence the link should take you to a collection of different relevant materials for you to continue your research, into the topics you find relevant.

If you are a programmer yourself and want to look at videos/presentations I've found helpful: [here you go](#).

Chat Jeopardy

By now I've hopefully established that repetition is the minimum and the knowledge gained through it is ephemeral.

Also, engagement is critical to uncover your own lack of understanding and to fill in the foundation on which new information can rest.

Under those aspects, It becomes obvious that this thing is the most destructive thing you could ever use as a student.

As a student your job is to learn.

Even if the teacher gives you tasks to fail at, if you repeatedly use an LLM to answer the questions, thinking, that the repetitive reading of the answers will make you learn, you are setting yourself up for failing not in school, but in life.

You will learn nothing from reading a pre stated answer.

"But its like a dictionary in better".

Yes, if you want your dictionaries to lie to your face with high confidence, that is.

Please explain C pointers to me

Sure! Pointers in C are variables that store memory addresses. They are incredibly powerful and fundamental to understanding C programming. Here's a breakdown:

1. **Memory Address:** In C, every variable you declare is stored somewhere in the computer's memory. Each of these memory locations has a unique address.
2. **Pointer Variable:** A pointer variable is a variable that stores the memory address of another variable. Instead of directly storing a value, it stores the address where the value is stored.
3. **Declaration Syntax:** To declare a pointer variable, you use an asterisk (*) *before the variable name. For example:*
`int ptr;`

The **only** thing correct in the explanations given is the syntax example.
Statement one and two are wrong.

Chat jeopardy is a little toy which is useless.
Meaning it has not a single applicable use.

But why will ChatGPT vehemently defend the definition it has given you?

source:

Maybe first, why the pointer answer from OAI was wrong:

The problem is that we execute code on a computer, which has an operating system.

So most often, if you are going through the standard library,
you are using the OS features such as [Virtual Memory](#).

Hence a pointer is simply something which lives one abstraction layer above the hardware.

There are ways of accessing memory directly which depend on the compiler and your target.

Same for point two. Most compilers are smart enough to substitute for the actual value at the right time.

So while one could argue that the `pointer = memory address` model is not completely wrong,
the model is wrong in all use cases a normal programmer encounters.

And if you are working in embedded software, you already know how to do it correctly.

Next is the part where I claim that LLM's are only [stochastic parrots](#).

Well, that's just what multi-layered neural nets are.

If you need an introduction to the field,
[this gentle introduction](#) got you covered.

Repetition

But in the end it all comes down to this.

Repetition.

The sources it was trained on repeated that misinformation multiple times,
so it is sure, that it is correct.

And the same way repetition of wrong facts propagates the myth that Java, C++ and C# are object oriented,
the same way all LLMs will propagate wrong information about almost every topic.

You will learn nothing by using a LLM.

Be excellent

I've decided to use my opportunity with this talk,
to enable the message to get across.

In the hopes that you remember what I said today every time,
you hear one of the other presentations that come after me.

That if you find anything meaningful in the things that get presented here,
that you then also take the time to download their transcription,
and read it again and again until you are able to understand what they have been trying to say.

You won't have learned something from this,
but maybe, maybe, you will remember something from this 15 minute presentation.