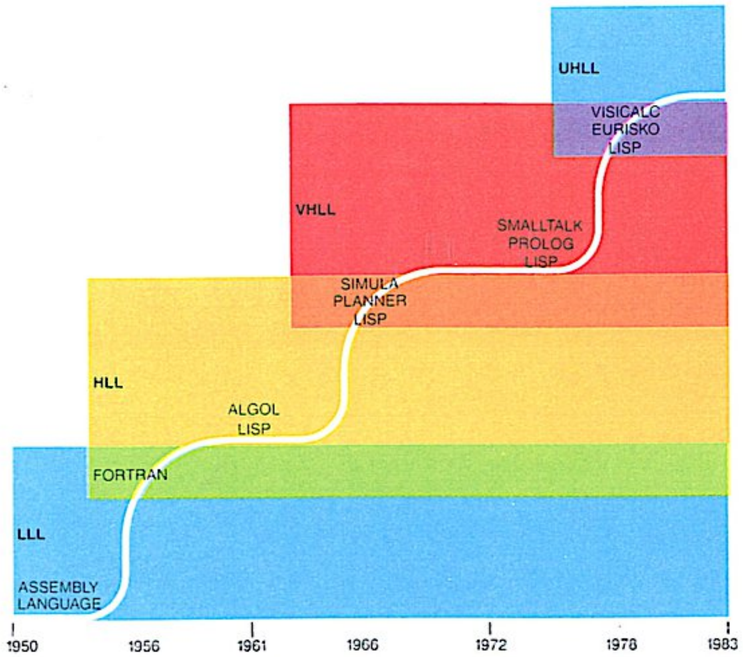
OaaS

Objects as a Service

$1T

Global business software and services market size by 2029[†]

Software development technologies aren’t progressing since 1980s



SOFTWARE GENRES succeed one another at sporadic intervals, as is shown here through the example of some programming languages. Languages are categorized rather arbitrarily by level, although the levels (*colored bands*) overlap. There are low-level languages (LLL), high-level languages (HLL), very-high-level languages (VHLL) and ultrahigh-level languages (UHLL). In the evolution of programming languages a genre is established (*horizontal white lines*), then after a few years an improvement is made (*curved white lines*). In time the improved language is seen to be not merely a “better old thing” but an “almost new thing,” and it leads to the next stable genre. The language Lisp has changed repeatedly, each time becoming a new genre.

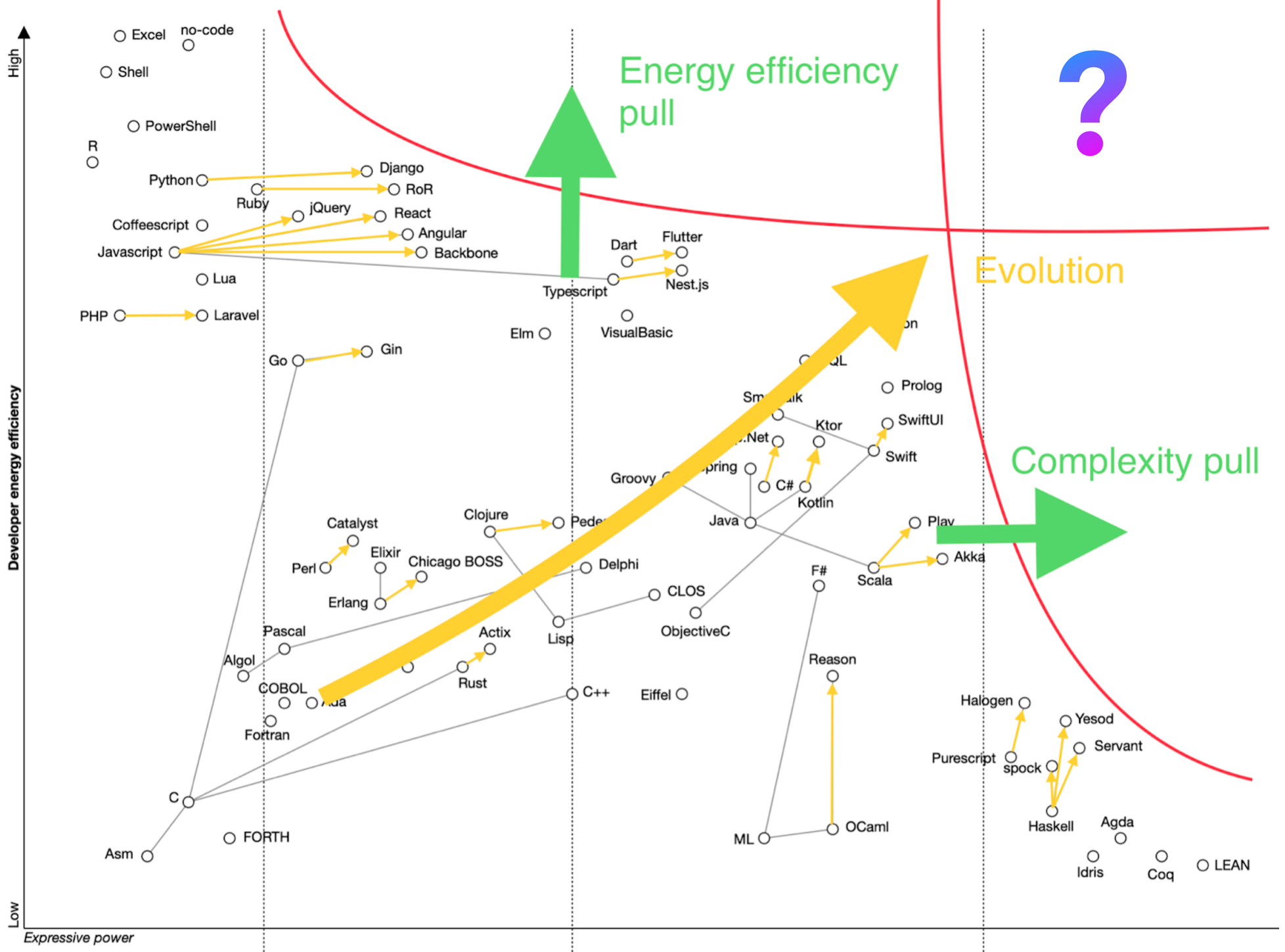
Current generation tools reached universality, by horizontally scaling with humans instead of further scaling tech vertically, and stopped evolving.

Current no-code is just fragmenting the space by drill-downs into verticals, doesn’t provide a qualitative solution to build up.

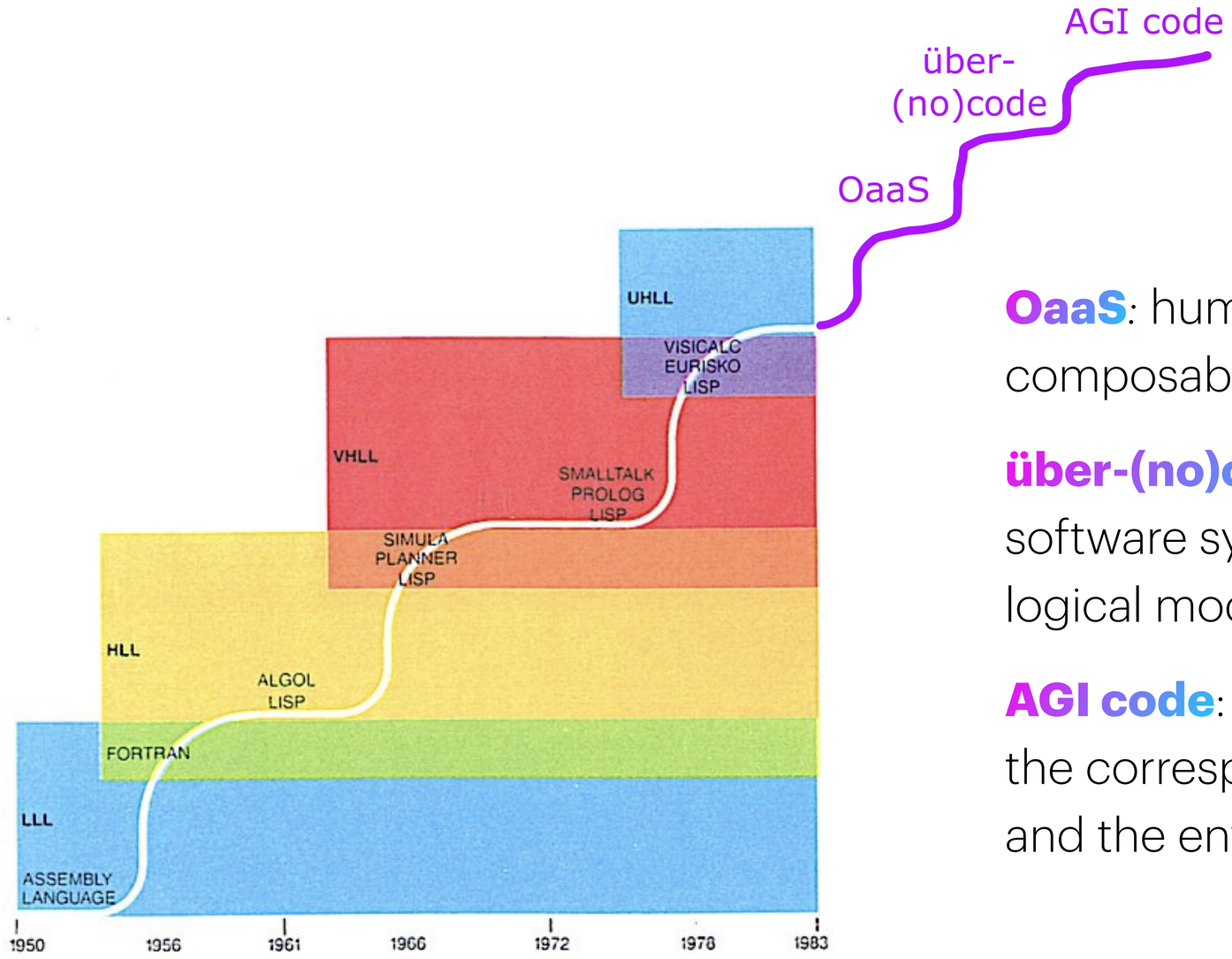
www.quora.com/What-was-the-last-breakthrough-in-computer-programming/answer/Alan-Kay-11

Evolution model

traversable.space/#essays/guts-d



Future of Software Development



OaaS: humans build complex software systems using composable objects and no-code tools.

über-(no)code: humans+*AI assistants* build complex software systems using human language to specify logical models, constraints and attractors.

AGI code: AGI builds and updates logical models and the corresponding software by interacting with humans and the environment.

Why OaaS

Current software is built with *Algorithms&Data structures* programming model. Its constructs are not powerful enough to model evolving systems. It works while doing **simple systems**. These systems are rigid, non-flexible, do not scale to evolution.

Also, current software is creating increasingly more **its own complexity** than the actual business functionality. It is more like pyramids rather than evolving organisms.

Real-world business applications, evolving in time and scale, and interacting with the external environment, quickly become **complex systems**.

This is where the **impedance mismatch** comes from. We need an "**organizing complex systems**" model.

"Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves."

— Alan Kay.

Objects as a Service FTW!

OaaS provides an environment and a UI which let users specify:

- ***business entities,***
- ***business processes,***
- ***connections,***
- ***integrations,***
- ***constraints and required states,***
- ***business rules.***

Users build the desired functionality by creating objects, or reusing existing ones from the library, and by connecting these objects into processes, pipelines, networks.

OaaS manages the implementation details:

infrastructure, scalability, persistence, data layer, messaging, protocols, evaluation strategies, dynamic dispatch, AAA, logging, metrics, resilience, availability, UI composition, testing, integrations, messaging, routing, deployment, hosting, CI/CD, version control,

backup and recovery, environment, sessions, contexts, state machines, distributed transactions, IDE, programmatic UI, SAT constraints solver, configuration, state history and rollback.

Perfect match with the Cloud

Objects live in the cloud, are universally addressed, communicate natively.

API integrations are just objects communication.

Objects can be stored in an inventory, can be bought and sold on a marketplace.

This was a vision of inventors of OOP at PARC, and then of Steve Jobs.

Unfortunately, they didn't have cloud and market ready.

Fortunately, we do have cloud and market now.

This is a next step after cloud functions, such as AWS Lambda or Azure functions.

Functions are not powerful enough to simulate business entities and relations, objects are.

OaaS is...

– A.Kay's OOP ✕ nocode ✕ cloud ✕ zero maintenance

– post-DevOps, post Kubernetes, post containers, post-OS

– post-IaaS/PaaS/FaaS/SaaS

– a medium for sculpting software to business requirements, in dynamics

– for business domain experts, not programmers

– building software like the Internet.

"The best way to predict the future is to invent it."

— Alan Kay.

$$$ → R&D → OaaS

Everyone able to step-by-step build a logical domain model is able to build software.

The industry is expanded by 10sX

Value is created

PROFIT!

WIAN – What Is Actually Needed

"If you want to make money, don't make a startup, start an industry."

"Instead of innovating out from the present, what you want to do is invent the future from the future. Go out and live in the future and bring the future back."

"Computer science inverts the normal. In normal science, you're given a world, and your job is to find out the rules. In computer science, you give the computer the rules, and it creates the world."

— Alan Kay.