Graffiticode

The Big Ideas

Prefix

Graffiticode began as an experiment in turning words into art – exploring how language can transmit meaning across the semantic boundary with visual form.

That experiment grew into a tool for domain experts of all kinds to create software using task specific code.

Fast forward to now, these domain specific languages provide the ideal target for generative AI as well as humans. The deterministic formal language is the perfect partner for the probabilistic language models.

Graffiticode is for domain experts to make μApps with the help of the latest generative language technologies.

Ideas

1. µApp as ubiquitous surface
2. Language oriented programming
3. End-user developers
4. Task specific languages
5. Natural language interfaces
6. API first
7. Pay-as-you-go
8. Cloud hosted
9. Open source everything
10. The infinite game

1 μApp as ubiquitous surface

For the same reasons that graffiti writers choose to express themselves on city walls for their uniformity, ubiquity and accessibility, we choose μApps on the open web as our canvas.

μApps are niche web apps that include an embeddable frontend supported by a dedicated backend. Graffiticode is a cloud based tool for creating μApps with application specific programming languages in the middle.

By combining web apps and domain specific language we have a universal framework with which to express an unlimited number of secure, scalable and ubiquitous software solutions.

μApps are an ideal canvas on which to express ideas in the web.

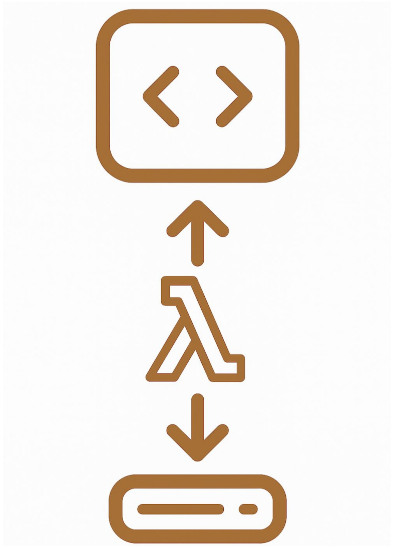
2 Language oriented programming

Language is the ultimate creative interface.

Martin Ward’s 1994 paper *"Language Oriented Programming"* highlighted how domain-specific languages let humans reason in application semantics rather than machine semantics.

Graffiticode extends Ward's vision to include generative AI as a consumer of the application language.

The compiler for a language provides feedback to both human and machine code generators to guide the software development process.



3 End-user developers

Software succeeds when the people who know the problems build the solutions.

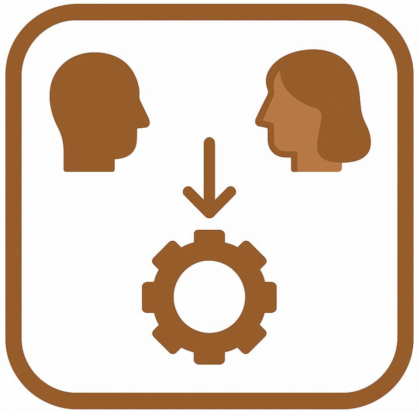
Traditional development marginalizes domain experts, leaving blind spots in the development process.

In the era of AI, domain experts will have unprecedented power to create value for business and society.

Graffiticode flips this model:

- Subject-matter experts build apps directly, writing in a vocabulary they already understand
- System developers craft the underlying languages and frameworks for domain experts to use

The language in the middle lets designers, educators, and practitioners of all kinds create and ship solutions of their own designs.



4 Task specific languages

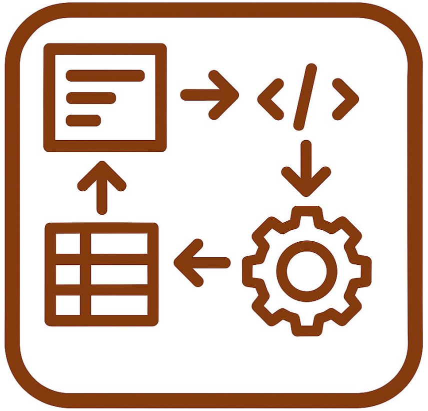
Each Graffiticode app has a task specific language at its core.

That language is used to express the state of the application:

- **Form frontend** captures user actions and emits code
- **Compiler backend** converts that code into a data model to be rendered by the form

This **Form** → **Code** → **Compile** → **Data** loop keeps the data model consistent with the app's live state at all times.

The formal language in the middle serves as a guard rail for the code generator, whether human or machine. By providing corrective feedback, the compiler helps the programmer write correct code.



5 Natural language interfaces

Task specific languages make a target rich environment for language models.

Graffiticode languages with their domain vocabularies and compilers take load off the language model to know what is true.

The Graffiticode toolchain gives LLMs the language surface and the feedback needed to generate error free and accurate translations of user requests.

Language runtimes written by system programmers ensure that generated apps are scalable and secure.

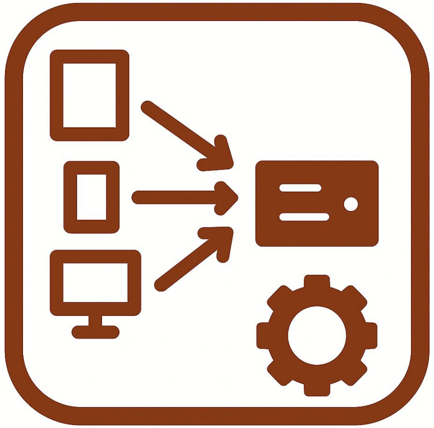
For any given user request, the response is always a complete, scalable and secure µApp.

6 API first

Every Graffiticode µApp is API-first. One backend can power multiple frontends. A Graffiticode dialect defines both the API to the backend and the model for the frontend.

One API for all stages of the application lifecycle. During development the editor calls the API to create the initial state of the app. During runtime the app calls the API to update and record the application state.

Backends are backward compatible always and forever. Every Graffiticode dialect has only own major version. Code that compiles compiles and runs forever.



7 Pay-as-you-go

The platform encourages usage based billing.

- Backend work is metered in **compile units**. A single request may trigger several compiles. Complex compiles may cost multiple units
- Compiles are **idempotent** – results are cached per codehash – so posting the same code twice is free

Whether you self-host or use a Graffiticode managed host, these economics align your costs with the value you get.

8 Graffiticode . org

graffiticode.org (GC.ORG) is a community hosted instance of Graffiticode.

GC.ORG is an out-of-the-box hosting of the Graffiticode project. It is supported by user payments and donations. The goal is to return all income to the project and its contributors.

GC.ORG supports for-profit third party languages through its language marketplace. Language developers get paid by subscription and/or usage.

Competing hosting services are welcome and encouraged. We want the Graffiticode ecosystem to grow. We see graffiticode.org as a model to be copied.

graffiticode.org provides production hosting of the console, API gateway and various languages.

9 Open source everything

Graffiticode's stack is 100% open source, MIT-licensed, and free to use.

By putting the core code in public repos we unlock four key advantages:

- **Community expertise** – anyone can audit, fork, and improve the code
- **LLM training data** – foundation models can train on the underlying source code and documentation, boosting both system and app intelligence
- **Shared IP, fewer disputes** – what's open for one is open for all, reducing ownership friction
- **Code transparency** – clarity about how the platform works builds lasting trust

The value of open source is in the ability to freely read and share code. The value of code is in the next commit not in the last one.

10 The infinite game

Graffiticode plays not to *win* but to keep improving the *play*.

James P. Carse defines an *infinite game* as one played “for the purpose of continuing the play.” We embrace that ethos:

- Rules evolve – new languages, new runtimes, new protocols are possible and encouraged
- Language experiments are ubiquitous. Failure is localized to each experiment. Success makes the global game better
- Our eternal north star is to make the platform more playful and productive for the next move and new players

"There is but one infinite game." – [James P. Carse, *Finite and Infinite Games*, 1986]

The Graffiticode organization is committed to keeping this small part of the infinite game alive and well.

Postfix

Come play with us

- Join the community forum at https://forum.graffiticode.org
- Create an account at https://graffiticode.org
- Get the code at https://github.com/graffiticode
- Share these slides
- Make a language