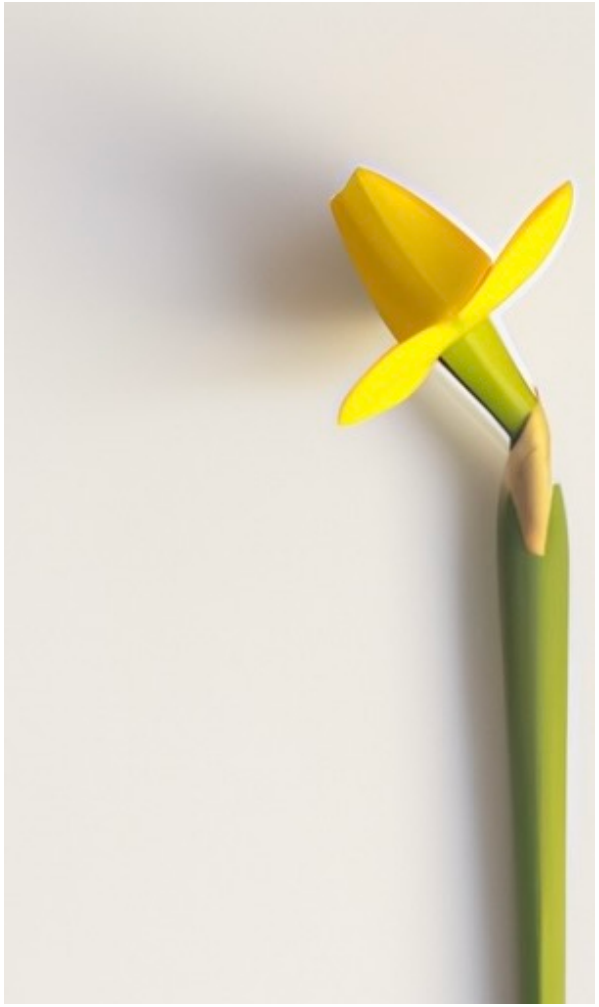
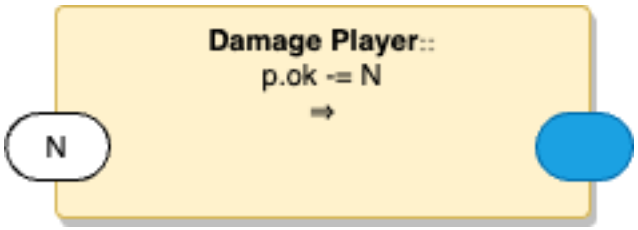
Design vs. Type Checking



What I Want to Say



What I Have to Say Instead

Type Checking Obfuscates Design

- Including enough niggly details for commanding a dumb machine makes code look “too busy” to humans.
- The Designer wants to say “N”, with the implication that N is an int and arrives as part of the incoming message.
- Instead, the Designer is forced to write detailed gobbledy-gook to appease the type checker, such as `msg.datum.data.(int)`
- The Designer should be allowed to express the Design in a concise manner, then, punt to a Production Engineer to add sufficient details
- Provenance must be maintained. Whatever the Production Engineer does, must be tracked back to what the Designer wrote/intended, in some way.
- There should be multiple views of the code - the Design View, and, the Production Engineering View.
- How can we accomplish this?

Lisp and Macros

- Lispers solve this kind of problem by creating *macros* to hide-away overly-busy details.
- Lispers don't *delete* the details, they just mask the details.
- In essence, a Lisper can view the code in 2 ways (1) view the original code with details hidden behind macros, and, (2) view the code with macros expanded, to see all of the niggly details.

Functional Programming vs. Macros

- Functional programming essentially results in macros - a mapping of text to some other text.
- All of the "rules" of Functional Programming support creating maps, e.g. "no side effects", "referential transparency", etc.
- But, the "rules" of Functional Programming bring ad-hoc baggage into programming, like ad-hoc blocking (a function blocks when it calls another function), hard-wiring names of other functions into code at the call point, ad-hoc routing decisions (a called function must always route its result back to the caller), etc.

Macros for Non-Lisp Languages

- PEG - Parsing Expression Grammars - can be used to create macros for non-Lisp languages.
- Lisp macros work on Lisp lists, whereas PEG macros work with characters and text, i.e. more modern languages.

Can Macros Be Used to Map Between Design Views and Production Engineering Views?

- Maybe, we want to say p -= N, where N = `msg.datum.data.(int)`?
- Would this be a step towards solving the Design vs. Production Engineering code disparities?
- Can we hide the *where* clause, so that Designers don't have to see it nor specify it?
- Can we maintain provenance? I.E. can we switch between Design View and Production View automatically?

Programming Simplicity

ohmjs.org

See Also

References *https://guitarvydas.github.io/2024/01/06/References.html*

Blog *https://guitarvydas.github.io/*

Blog *https://publish.obsidian.md/programmingsimplicity*

Videos *https://www.youtube.com/@programmingsimplicity2980*

[see playlist "programming simplicity"]

Discord *https://discord.gg/Jjx62ypR*

X *(Twitter) @paul_tarvydas*

More writing *(WIP): https://leanpub.com/u/paul-tarvydas*