

System Injection

Side effect management in Pen

[@raviqqe](#)

What is System Injection?

- Pen has a feature called System Injection.
- It manages side effects of functions.
- It injects system functions into whole applications.
 - System functions are passed as arguments of `main` functions to applications.
- No other functions can have side effects by themselves; side effects are always injected through function arguments.

What is it based on?

- Dependency injection
- Monads
- Effect systems
- Clean architecture

Benefits of system injection

It improves maintainability and portability of software.

- Unit tests are deterministic.
 - No more slow or flaky tests!
- Better software architecture
 - Application logic doesn't depend on implementation details directly.
 - They are decoupled and changeable independently.
 - It's easier to port applications isolated from platform-dependent codes.

Costs of System Injection

- Extra cognitive load
 - Enforcement of dependency injection
 - Context arguments
- In the short term, it doesn't pay for the cost...
 - More like investment
 - The same as unit tests, or any other software engineering methodologies
 - Pen is not for scripting, or software with short expectancy in general.

```
print = \(\ctx Context, s string) none | error {  
  File'Write(ctx, File'StdOut(), s)?  
  
  none  
}
```

Common questions

How do we run nondeterministic tests?

1. Replace nondeterministic codes with deterministic codes.
2. List up examples as separate tests.
 - In other words, choose the *timelines*.

Common questions

Why not effect systems (or monads)?

- Modern researches propose "statically provable" effect systems.
 - As extensions of type systems
- They incur extra cognitive costs for developers to understand and use it.
- System Injection is rather a *dynamically typed* effect system.
- Also, they do not work well with programming languages without generics.
 - Like Go
 - High-order functions always need to have two versions for pure and impure implementations.
- [Functional programming | Clojure](#)
 - Clojure is impure, in that it doesn't force your program to be referentially transparent, and doesn't strive for 'provable' programs.

Running integration tests

- Deterministic tests are good but that doesn't mean we don't need to test E2E.
- Currently, Pen has no way to run integration tests.
- `pen test --integration ?`
- It's also possible to delegate them to third party tools.

Summary

- System Injection gives us:
 - Deterministic, fast, reliable unit tests.
 - *Clean* software architecture
- Integration tests are WIP.
- Feedback is welcome!