



Using Arch-Go to continuously test the quality of our architecture

Francisco Daines

About Me

Francisco Daines

- Software Developer - Walmart Chile
- Experience in Java, C, Javascript
- Gopher since 2020



fdaines



fdaines



fdaines

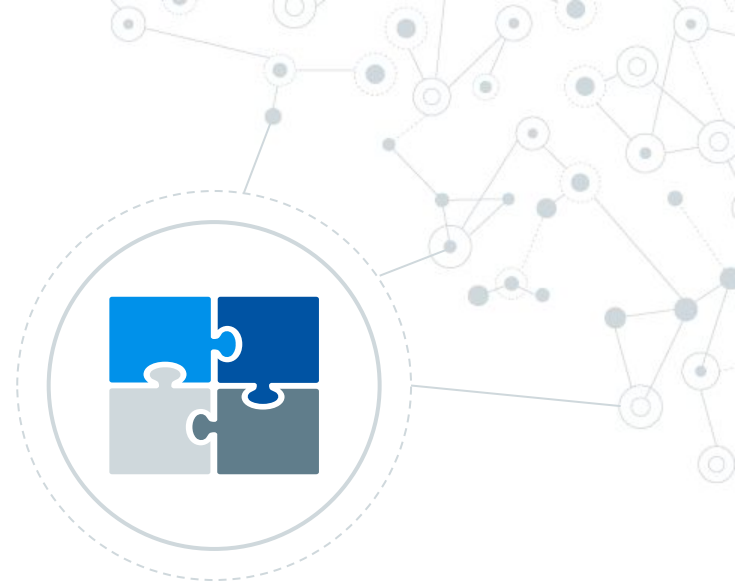


fdaines@gmail.com



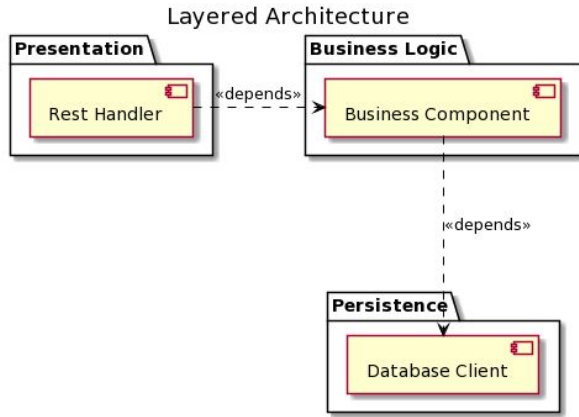
Architectural Guidelines

A set of important “things”



Architectural Guidelines - Packages Model

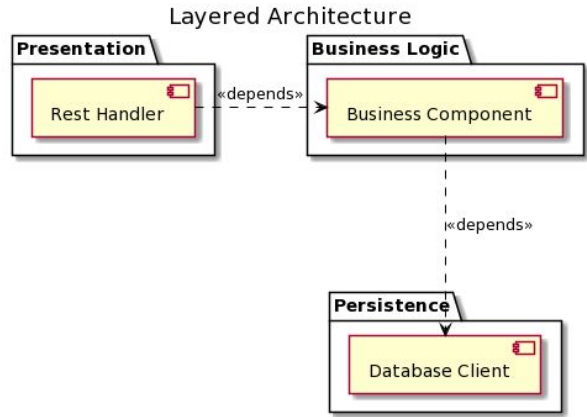
Choosing between different options, some examples



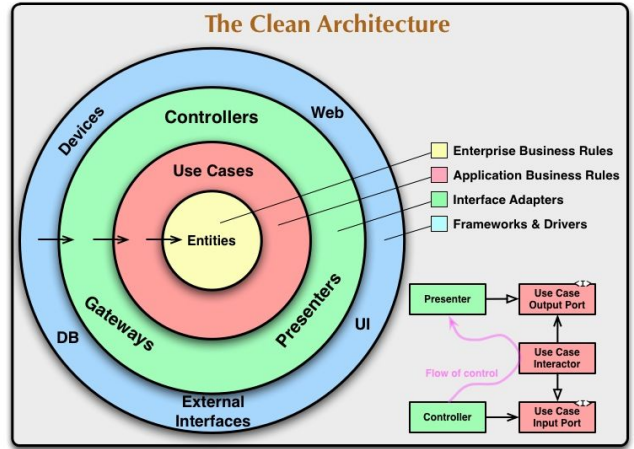
Typical Layered Architecture

Architectural Guidelines - Packages Model

Choosing between different options, some examples



Typical Layered Architecture



The Clean Architecture - Robert C. Martin

Architectural Guidelines - Other important agreements

Do we need some rules about

- How to create functions?
- Names of our items?
- What kind of items can be placed in certain packages?
- What are the allowed relations between packages?



Issues related to Architectural Guidelines

Versioning

As architectures can evolve, we need to record changes to our guidelines.




Issues related to Architectural Guidelines

Versioning

As architectures can evolve, we need to record changes to our guidelines.

Outdated Guidelines

Confluence (or similar) are far from code, then are prone to be outdated.





Issues related to Architectural Guidelines

Versioning

As architectures can evolve, we need to record changes to our guidelines.

Outdated Guidelines

Confluence (or similar) are far from code, then are prone to be outdated.

Software Quality Degradation

Even if practices like code reviews and pair programming are powerful techniques, tested code is less error-prone.



Issues related to Architectural Guidelines

Versioning

As architectures can evolve, we need to record changes to our guidelines.

Outdated Guidelines

Confluence (or similar) are far from code, then are prone to be outdated.

Software Quality Degradation

Even if practices like code reviews and pair programming are powerful techniques, tested code is less error-prone.

Metrics!

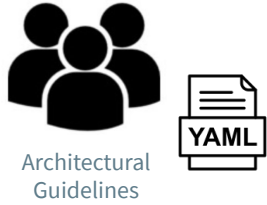
Is there a relation between the guidelines compliance level with other metrics?

Arch-Go

Getting Started

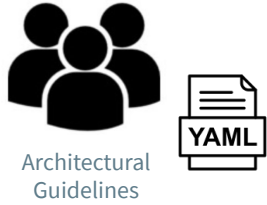


Verification Process Explained

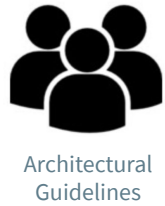


Verification Process Explained

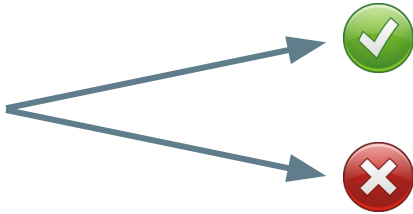
```
> go get -u github.com/fdaines/arch-go
```



Verification Process Explained



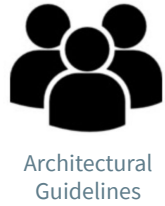
```
> go get -u github.com/fdaines/arch-go
> arch-go
```



```
INFO - Package **presentation** should only depend on: (**businesslogic)
PASS - Package **businesslogic** should only depend on: (**persistence)
PASS - Package **businessmanager** should not depend on: (**presentation)
PASS - Package **persistence** should not depend on: (**presentation **businesslogic)
-----
Total Rules: 4
Succeeded: 4
Failed: 0
Time: 0.720 seconds
```

Console output

Verification Process Explained



Verification

```
> go get -u github.com/fdaines/arch-go
> arch-go
```



```
INFO - Package **presentation** should only depend on: (**dependencies)
INFO - Package **businesslogic** should only depend on: (**persistence)
INFO - Package **businessmanager** should not depend on: (**presentation)
INFO - Package **persistence** should not depend on: (**presentation)
-----
Total Rules: 4
Succeeded: 4
Failed: 0
Time: 0.220 seconds
```

Console output

```
> arch-go --html
```

Arch-Go Verification Report

Breakdown by Rule

Rule Type	Summary	Total	Succeed	Fail
DependenciesRule	3/3	3	3	0
FunctionsRule	0/1	1	0	1
ConstantsRule	6/6	6	6	0
CyclicRule	1/1	1	1	0
NamingRule	1/1	1	1	0

Report generated by Arch-Go v0.4.7

HTML Report

Features

What Arch-Go offers?



Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Rules Evaluation

Checks if your project complies with the rules defined in the YAML file.

All the rules are evaluated, so the result contains all the gaps.

Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Rules Evaluation

Checks if your project complies with the rules defined in the YAML file.

All the rules are evaluated, so the result contains all the gaps.

HTML Report

Arch-Go generates an HTML report with the evaluation result.

Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Rules Evaluation

Checks if your project complies with the rules defined in the YAML file.

All the rules are evaluated, so the result contains all the gaps.

HTML Report

Arch-Go generates an HTML report with the evaluation result.

Rules Description

```
dependenciesRules:  
- package: "**.cmd.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.impl.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.arch-go.*"  
  shouldOnlyDependsOnExternal:  
    - "github.com/fatih/color"  
    - "github.com/spf13/cobra"  
    - "gopkg.in/yaml.v2"
```

Rules Description

```
dependenciesRules:  
- package: "**.cmd.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.impl.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.arch-go.*"  
  shouldOnlyDependsOnExternal:  
    - "github.com/fatih/color"  
    - "github.com/spf13/cobra"  
    - "gopkg.in/yaml.v2"
```

```
dependenciesRules:  
- { package: "**.cmd.*", shouldOnlyDependsOn: ["**.arch-go.*"] }  
- { package: "**.impl.*", shouldOnlyDependsOn: ["**.arch-go.*"] }  
- { package: "**.arch-go.*", shouldOnlyDependsOnExternal:  
  ["github.com/fatih/color", "github.com/spf13/cobra", "gopkg.in/yaml.v2"]  
}
```

Rules Description

```
dependenciesRules:  
- package: "**.cmd.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.impl.*"  
  shouldOnlyDependsOn:  
    - "**.arch-go.*"  
- package: "**.arch-go.*"  
  shouldOnlyDependsOnExternal:  
    - "github.com/fatih/color"  
    - "github.com/spf13/cobra"  
    - "gopkg.in/yaml.v2"
```

YAML files

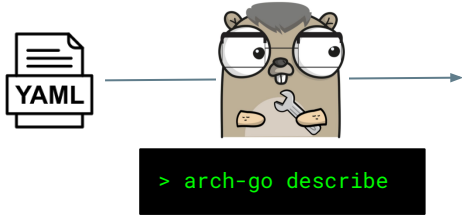
Both of these YAML configurations complies with the same schema.

```
dependenciesRules:  
- { package: "**.cmd.*", shouldOnlyDependsOn: ["**.arch-go.*"] }  
- { package: "**.impl.*", shouldOnlyDependsOn: ["**.arch-go.*"] }  
- { package: "**.arch-go.*", shouldOnlyDependsOnExternal:  
  ["github.com/fatih/color", "github.com/spf13/cobra", "gopkg.in/yaml.v2"]  
}
```

Rules Description



Rules Description



Dependency Rules

```
* Packages that match pattern '**.cmd.*',
  * Should only depends on packages that matches:
    - '**.arch-go.**'
* Packages that match pattern '**.impl.*',
  * Should only depends on packages that matches:
    - '**.arch-go.**'
* Packages that match pattern '**.arch-go.*',
  * Should only depends on external packages that matches
    - 'github.com/fatih/color'
    - 'github.com/spf13/cobra'
    - 'gopkg.in/yaml.v2'
```

Rules Description



```
> arch-go describe
```

Dependency Rules

- * Packages that match pattern '**.cmd.*',
 - * Should only depends on packages that matches:
 - '**.arch-go.**'
- * Packages that match pattern '**.impl.*',
 - * Should only depends on packages that matches:
 - '**.arch-go.**'
- * Packages that match pattern '**.arch-go.*',
 - * Should only depends on external packages that matches
 - 'github.com/fatih/color'
 - 'github.com/spf13/cobra'
 - 'gopkg.in/yaml.v2'



Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Rules Evaluation

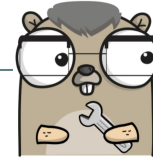
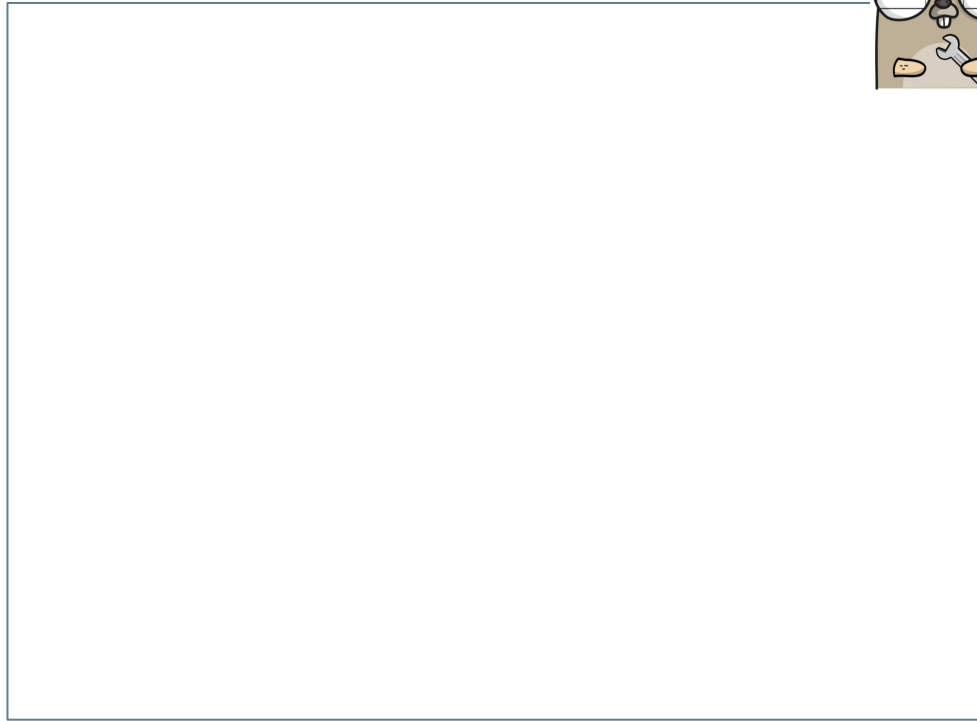
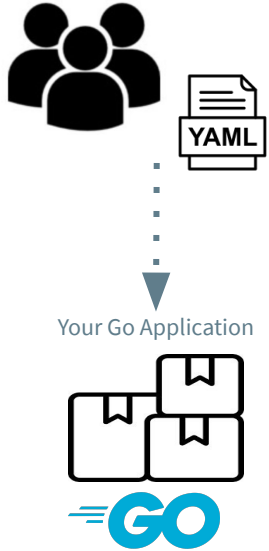
Checks if your project complies with the rules defined in the YAML file.

All the rules are evaluated, so the result contains all the gaps.

HTML Report

Arch-Go generates an HTML report with the evaluation result.

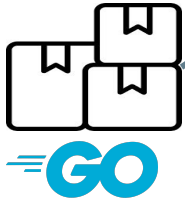
Rules Evaluation



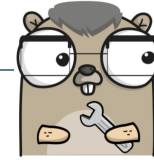
Rules Evaluation



Your Go Application



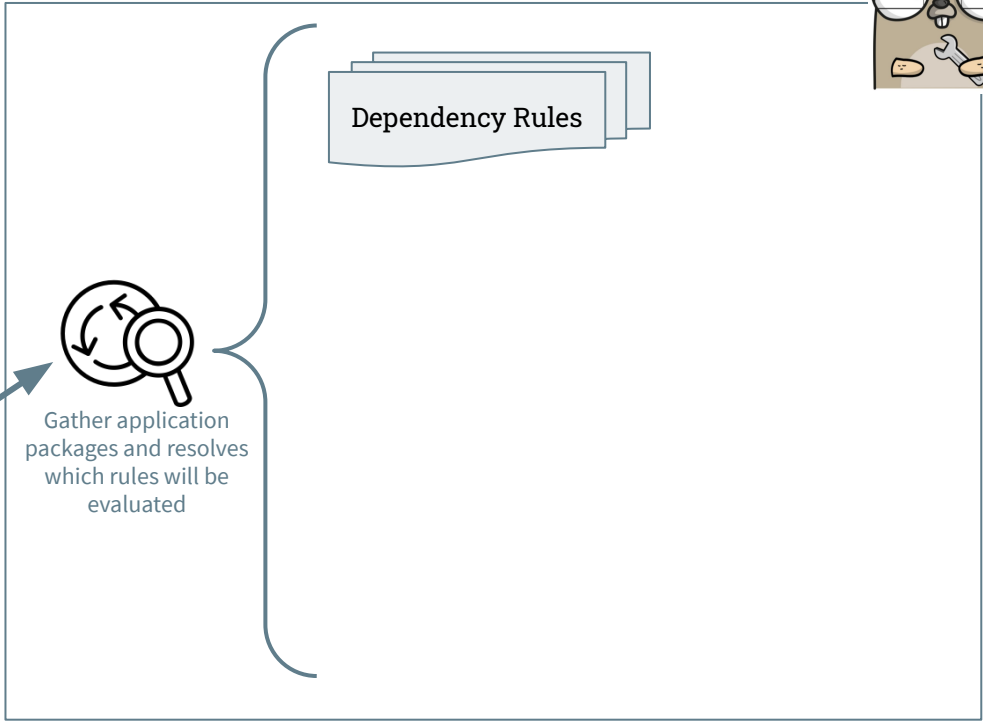
Gather application packages and resolves which rules will be evaluated



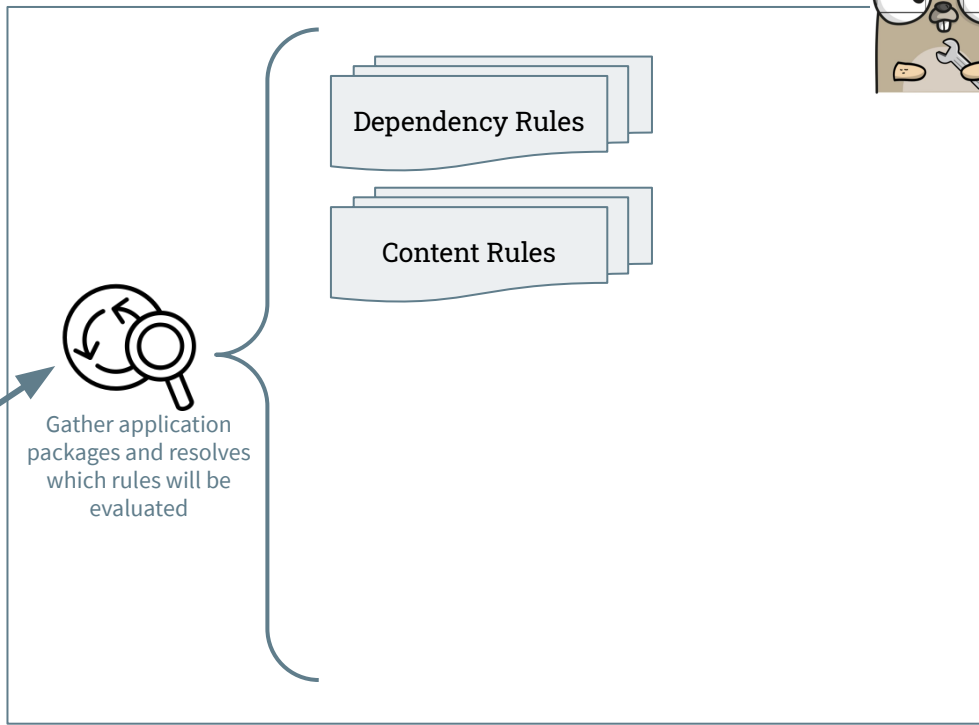
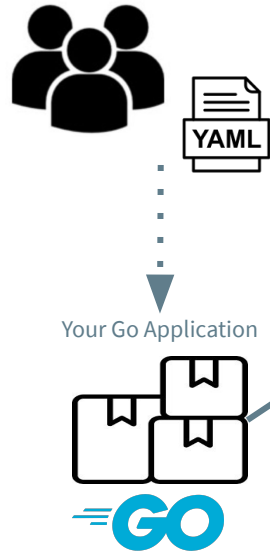
Rules Evaluation



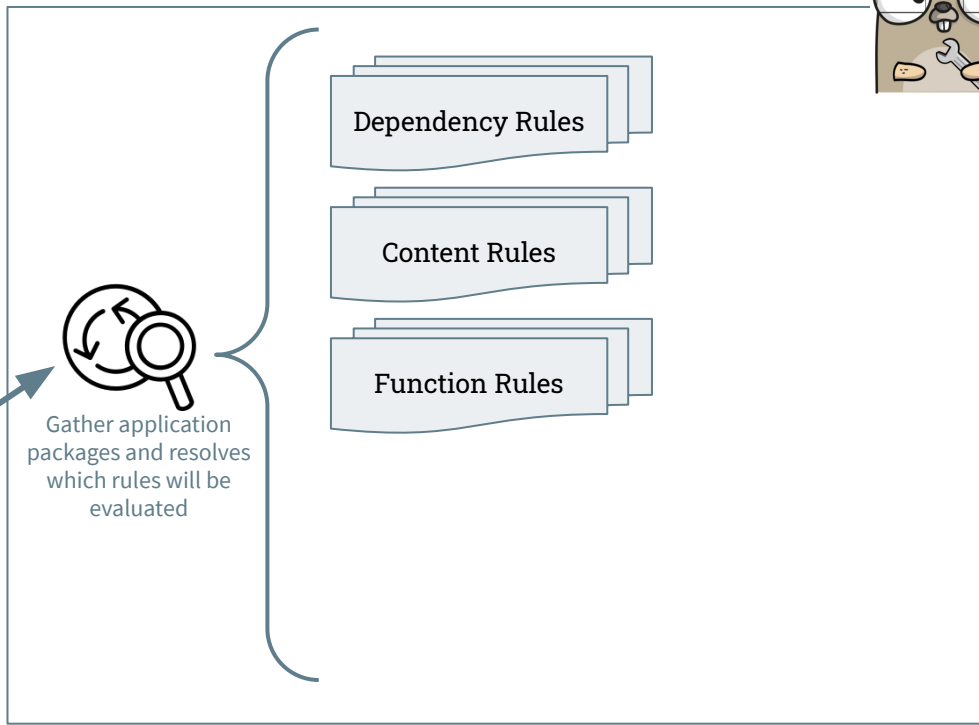
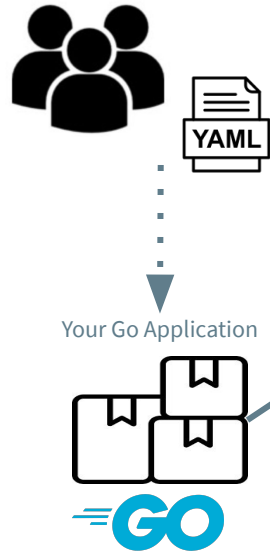
Your Go Application



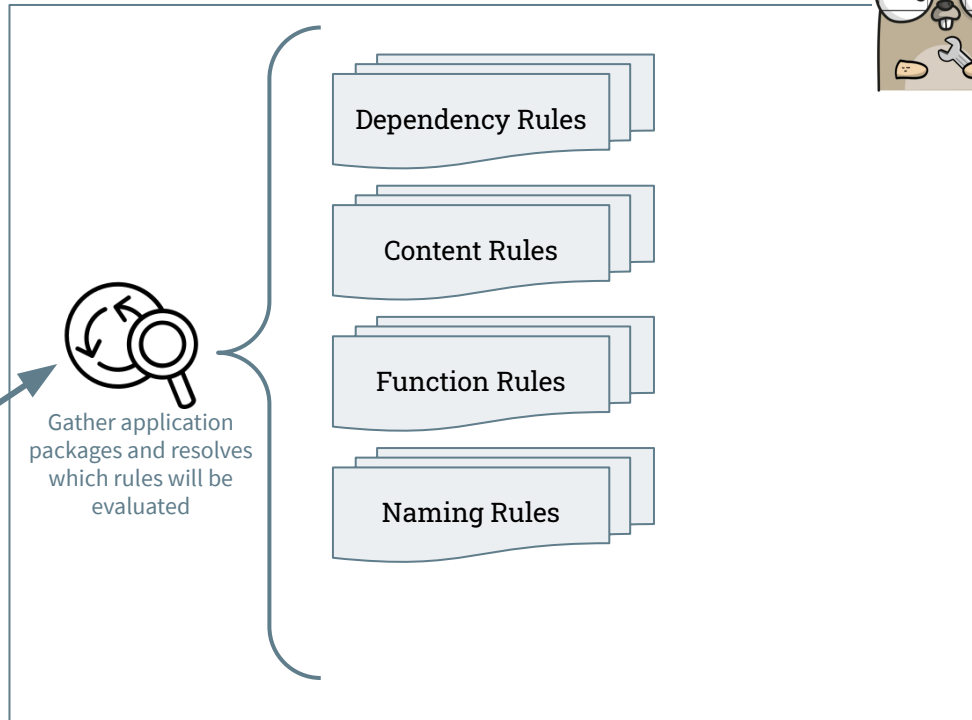
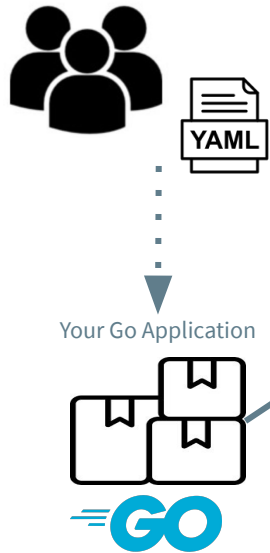
Rules Evaluation



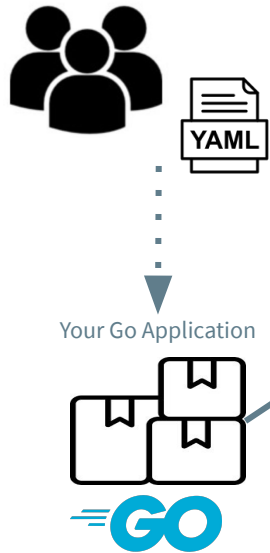
Rules Evaluation



Rules Evaluation



Rules Evaluation



Your Go Application



Gather application packages and resolves which rules will be evaluated

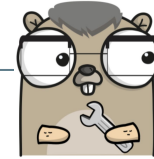
Dependency Rules

Content Rules

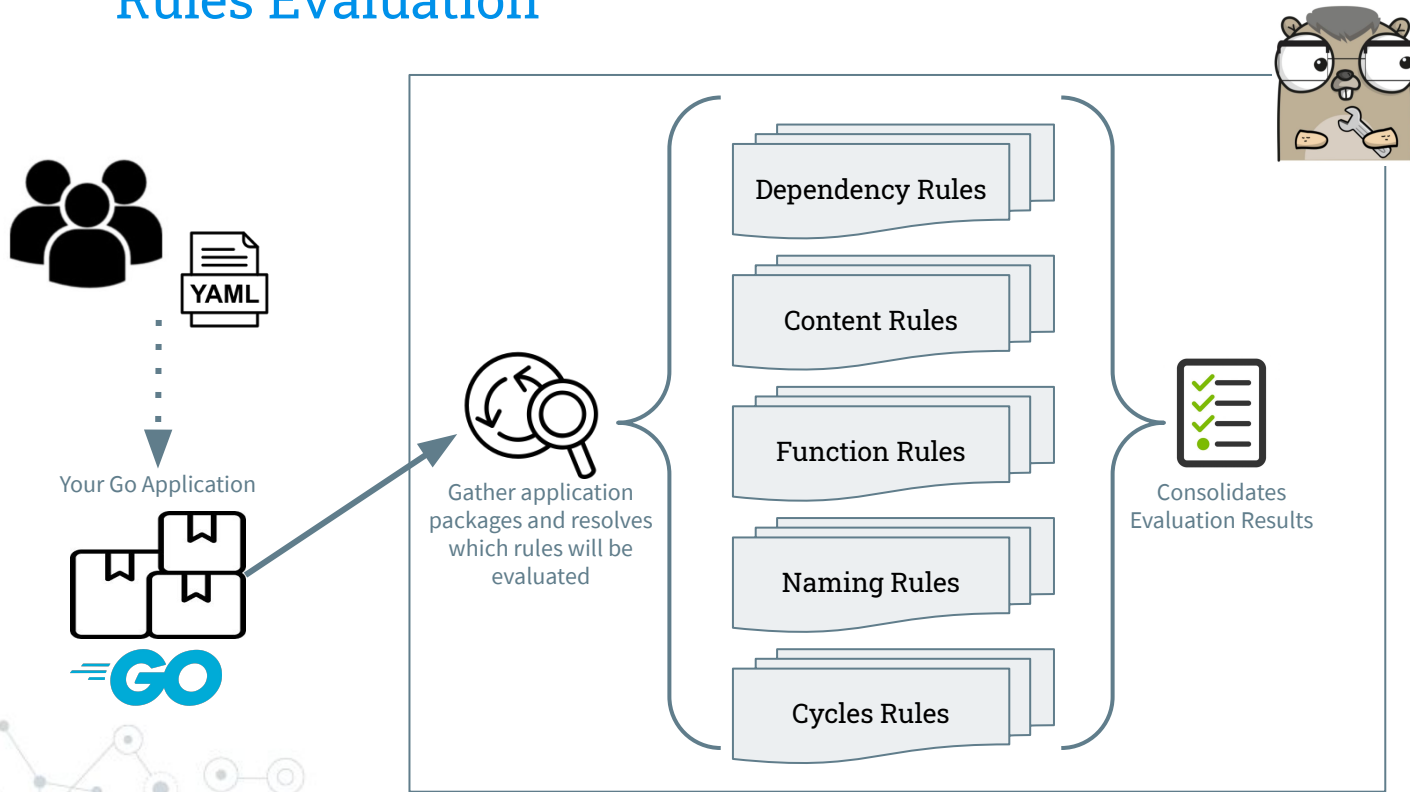
Function Rules

Naming Rules

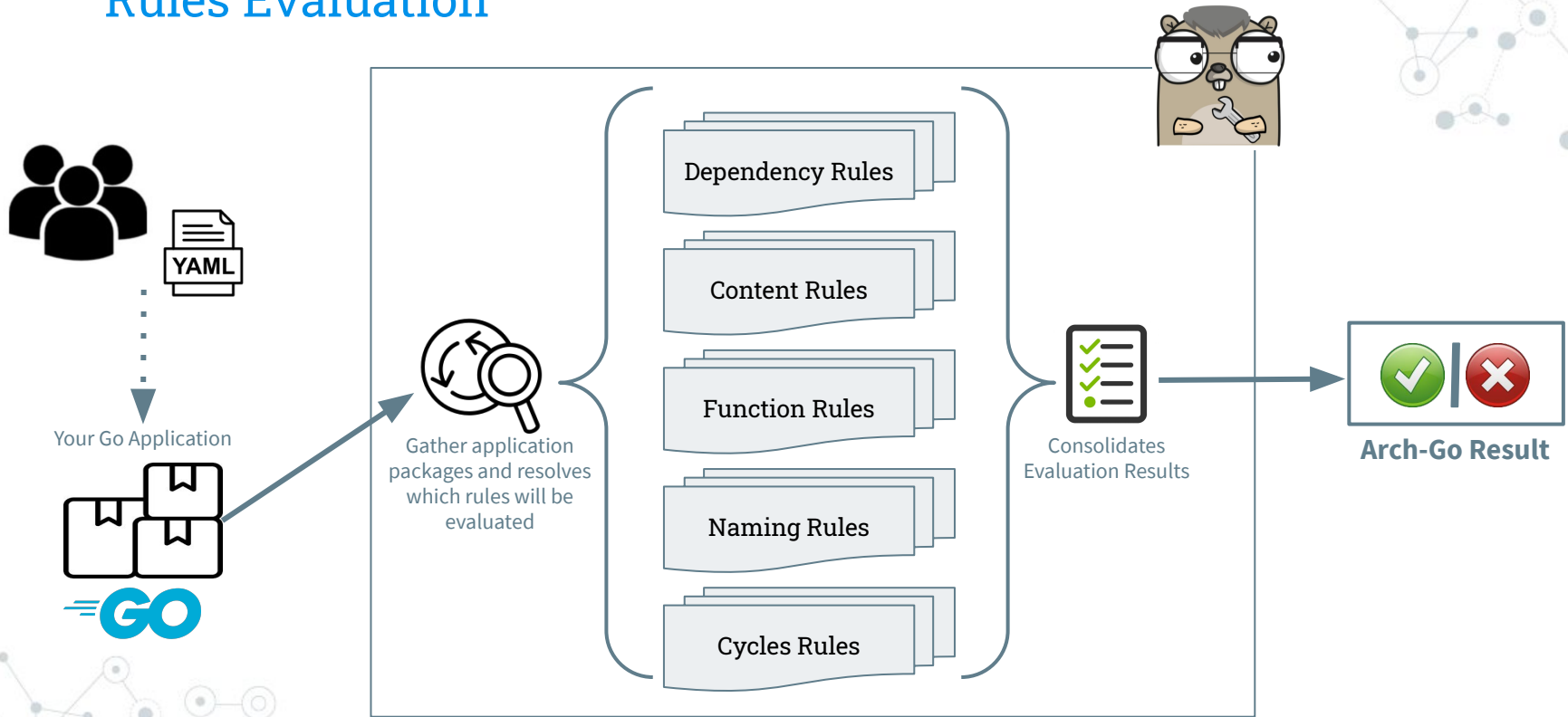
Cycles Rules



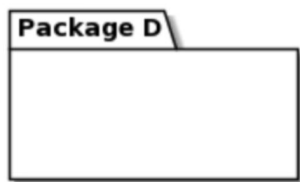
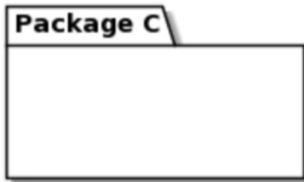
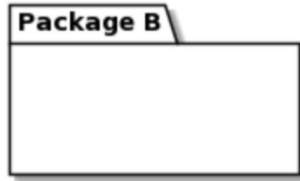
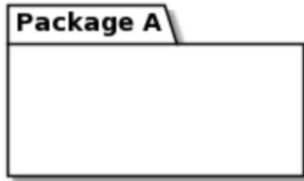
Rules Evaluation



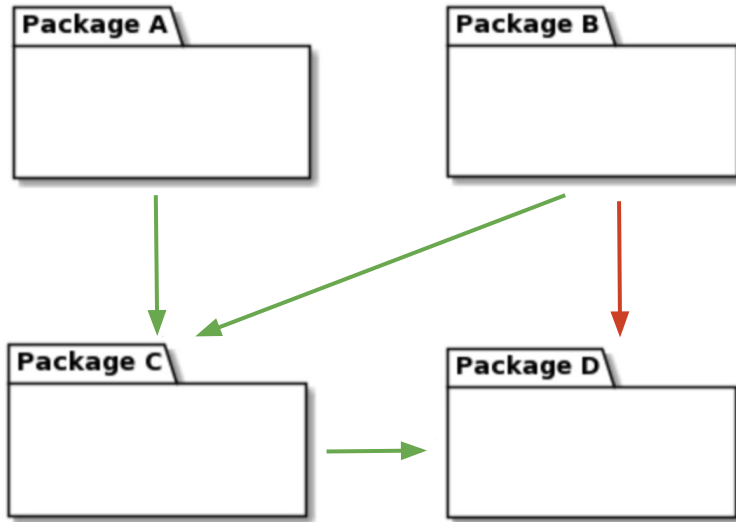
Rules Evaluation



Dependency Rules



Dependency Rules

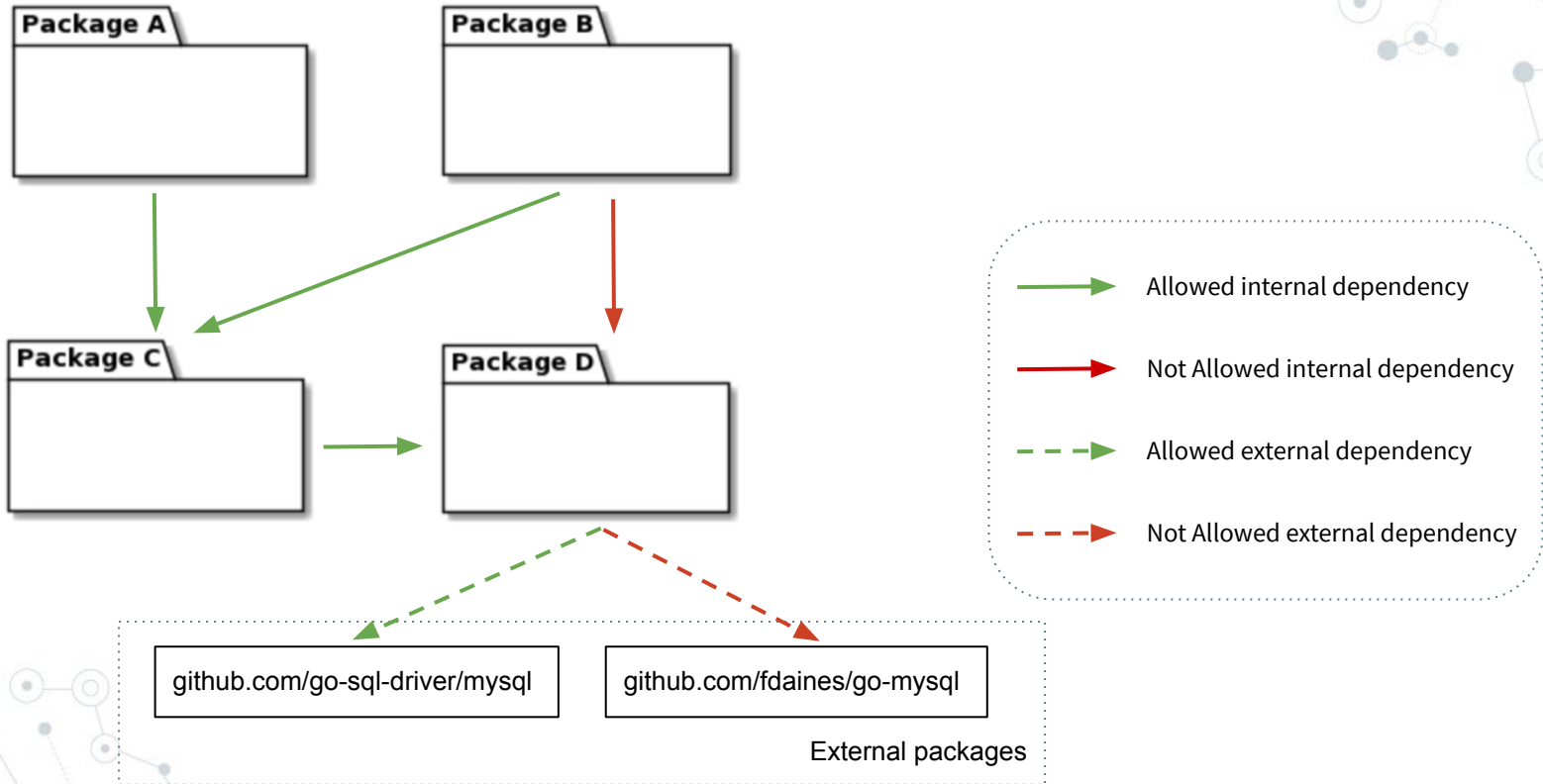


Allowed internal dependency



Not Allowed internal dependency

Dependency Rules



Dependency Rules

```
dependenciesRules:  
  - package: "**.cmd.*"  
    shouldOnlyDependsOn:  
      - "**.arch-go.**"  
  - package: "**.impl.**"  
    shouldNotDependsOn:  
      - "**.arch-go.**"  
  - package: "**.arch-go.**"  
    shouldOnlyDependsOnExternal:  
      - "github.com/fatih/color"  
      - "github.com/spf13/cobra"  
      - "gopkg.in/yaml.v2"
```


Content Rules

Package A



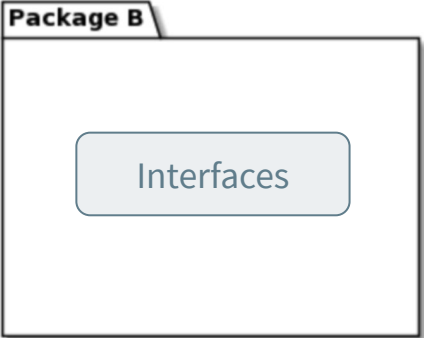
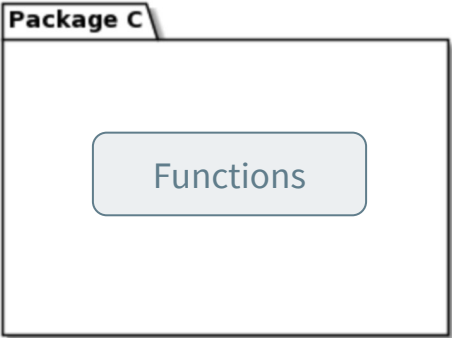
Package C



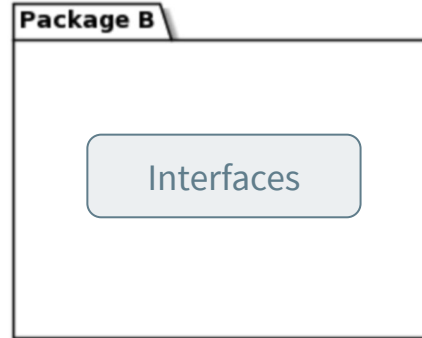
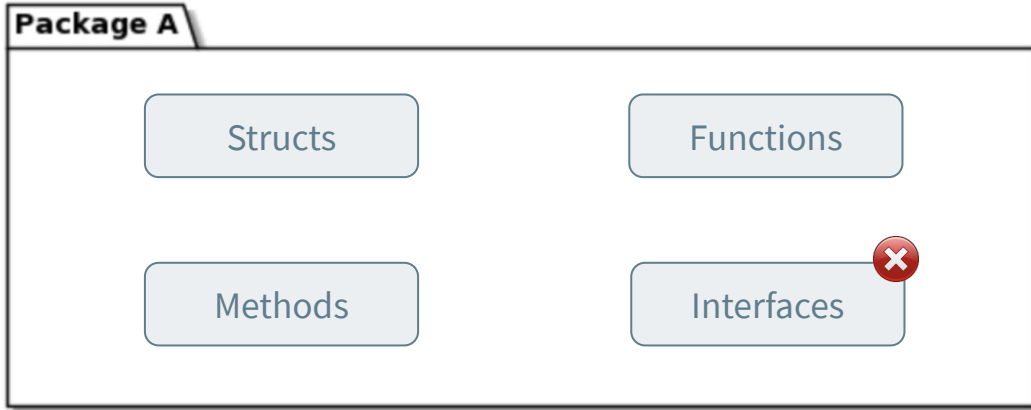
Package B



Content Rules



Content Rules



A method is a function with a receiver

Content Rules

```
contentsRules:  
- package: "**.impl.model"  
  shouldNotContainFunctions: true  
  shouldNotContainMethods: true  
- package: "**.impl.config"  
  shouldOnlyContainFunctions: true  
- package: "**.impl.dependencies"  
  shouldNotContainInterfaces: true  
- package: "**.impl.contents"  
  shouldNotContainInterfaces: true  
- package: "**.impl.cycles"  
  shouldNotContainInterfaces: true  
- package: "**.impl.functions"  
  shouldNotContainInterfaces: true
```

ShouldOnlyContain[ItemType]:

- Functions
- Methods
- Interfaces
- Structs

ShouldNotContain[ItemType]:

- Functions
- Methods
- Interfaces
- Structs

Function Rules

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```

Function Rules

Parameters quantity

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```

Function Rules

Parameters quantity

Return values quantity

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```

Function Rules

Parameters quantity

Return values quantity

Function size
(LOCs)

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```


Function Rules

Parameters quantity

Return values quantity

Function size
(LOCs)

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```

How many
functions per file
are allowed

Function Rules

Parameters quantity

Return values quantity

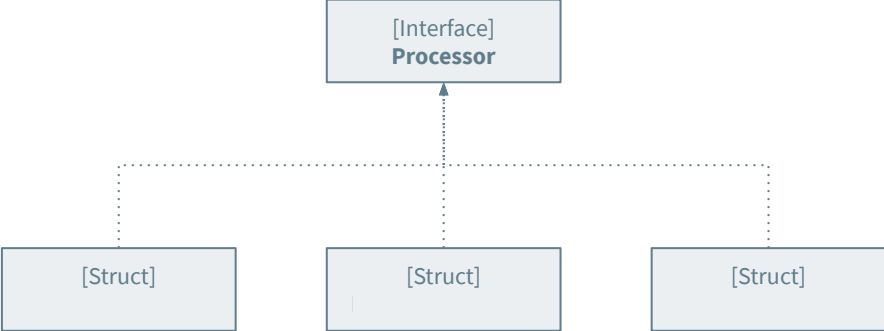
Function size
(LOCs)

```
func doSomething(param1, param2, param3 int) (string, string, err) {  
    ...  
    // do some stuff  
    ...  
    return "foo", "bar", nil  
}  
  
func doNothing() {  
}  
  
func sayHello() string {  
    return "Hello"  
}
```

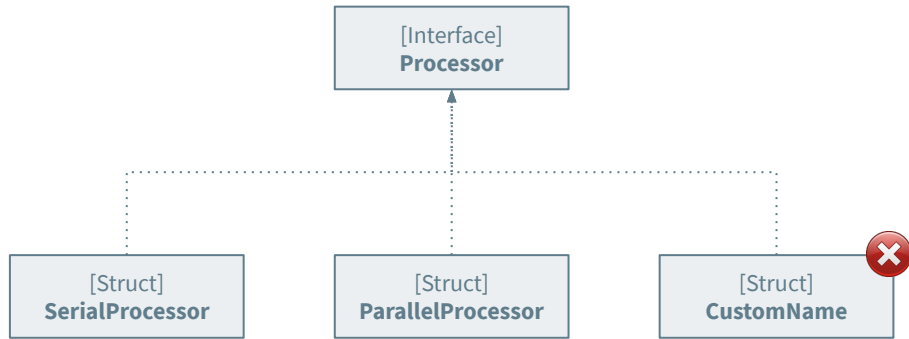
```
functionsRules:  
- package: "**.arch-go.**"  
  maxParameters: 4  
  maxReturnValues: 2  
  maxPublicFunctionPerFile: 5  
  maxLines: 50
```

How many
functions per file
are allowed

Naming Rules

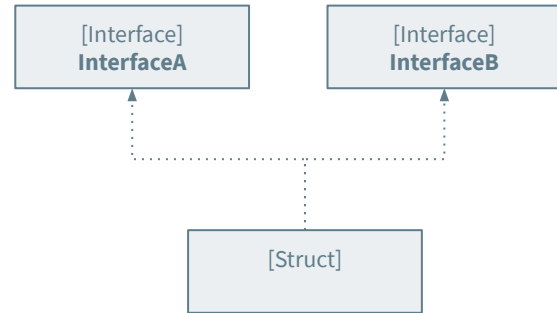
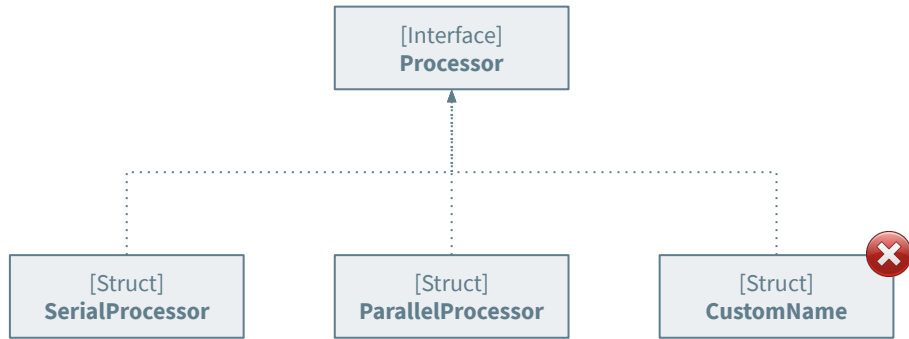


Naming Rules



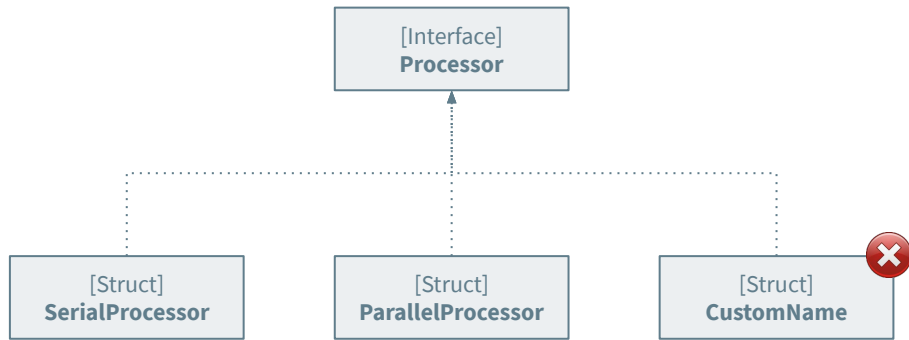
Sometimes makes sense to comply with a naming rule

Naming Rules

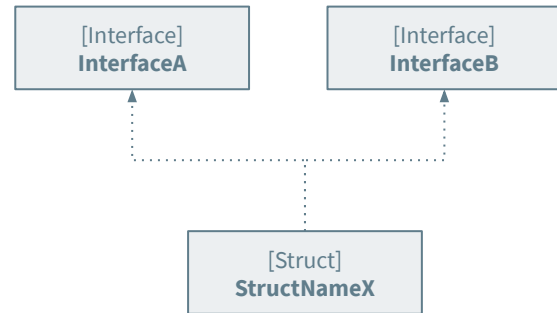


Sometimes makes sense to comply with a naming rule

Naming Rules

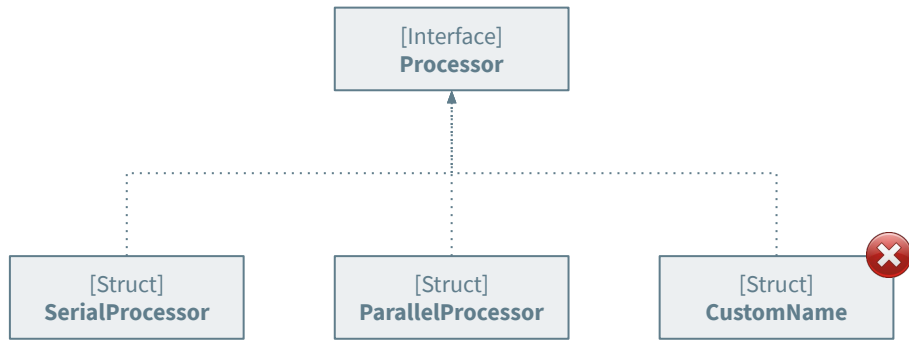


Sometimes makes sense to comply with a naming rule



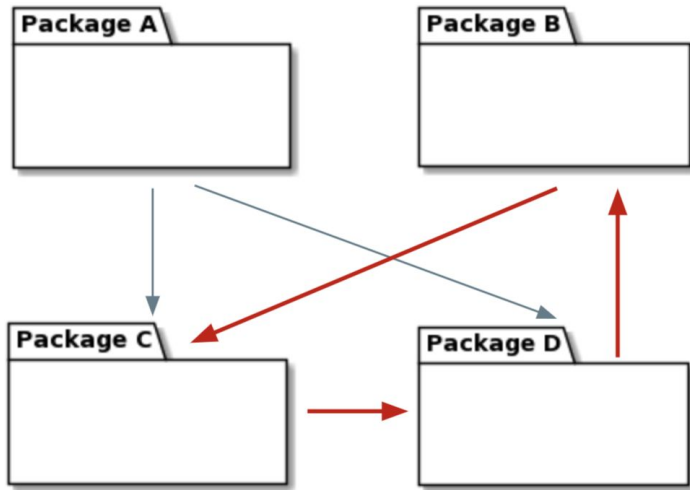
Sometimes not

Naming Rules

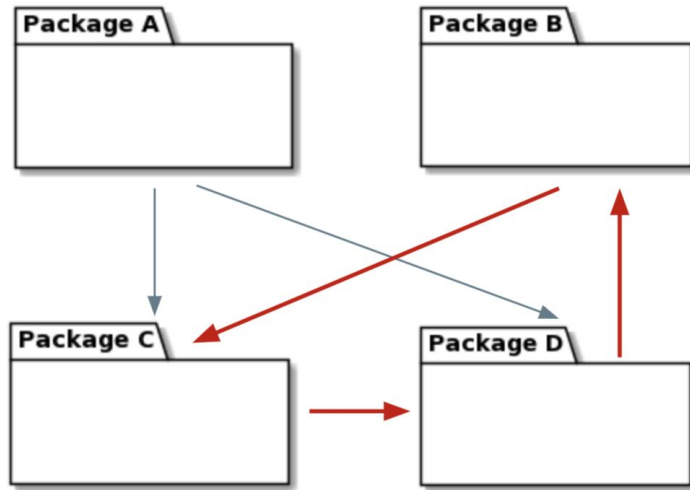


```
namingRules:  
- package: "**.arch-go.**"  
  interfaceImplementationNamingRule:  
    structsThatImplement: "*Processor"  
    shouldHaveSimpleNameEndingWith: "Processor"
```

Cycles Rules

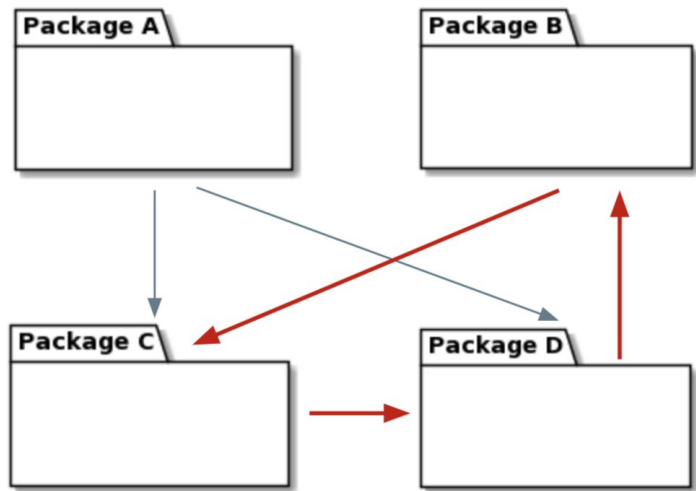


Cycles Rules



As Golang compiler does not allow import cycles, this rule option is under evaluation and maybe it will be deprecated in future releases.

Cycles Rules



As Golang compiler does not allow import cycles, this rule option is under evaluation and maybe it will be deprecated in future releases.

```
cyclesRules:  
- package: "**.cmd"  
  shouldNotContainCycles: true
```

Arch-Go Features

Rules Description

Architecture guidelines are defined in a YAML file.

Arch-Go can describe these guidelines using human language.

Rules Evaluation

Checks if your project complies with the rules defined in the YAML file.

All the rules are evaluated, so the result contains all the gaps.

HTML Report

Arch-Go generates an HTML report with the evaluation result.

HTML Report

Arch-Go Verification Report

Breakdown by Rule

Rule Type	Summary	Total	Succeed	Fail
DependenciesRule	3/3	3	3	0
FunctionsRule	0/1	1	0	1
ContentRule	6/6	6	6	0
CycleRule	1/1	1	1	0
NamingRule	1/1	1	1	0

Report generated by [Arch-Go](#) v0.4.7

Inspired on *PiTest Coverage Report*.

Work in Progress:

- *Navigate into evaluated rule details.*

Automation

What about the
“continuously” part?



Including Arch-Go as part of a CI/CD pipeline



Generic CI/CD Pipeline

Including Arch-Go as part of a CI/CD pipeline



Generic CI/CD Pipeline



CI/CD Pipeline with Architecture Tests

CI/CD - Examples



GitHub Actions

```
name: 'testing-pipeline'

on:
  workflow_dispatch:
  push:
    branches:
      - master

jobs:
  Arch-Go:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v1
      - uses: actions/setup-go@v2
        with:
          stable: 'false'
          go-version: '1.15'
      - name: Install Arch-Go
        run: go get -u github.com/fdaines/arch-go
      - name: Run Arch-Go
        run: arch-go
```



circleci

```
version: 2.1

jobs:
  build:
    working_directory: ~/repo
    docker:
      - image: circleci/golang:1.15.8
    steps:
      - checkout
      - run:
          name: Run Arch-Go
          command: |
            go get -u github.com/fdaines/arch-go
            arch-go
```



Bitbucket Pipelines

```
image: golang:1.15

pipelines:
  default:
    - parallel:
        - step:
            name: Running Arch-Go
            script:
              - go get -u github.com/fdaines/arch-go
              - arch-go
```


Want to Contribute?



Everyone is Welcome!

- <https://github.com/fdaines/arch-go>

Ideas Backlog

- Improve code coverage
- Validation of rule description file (YAML)
- Support for both YAML and JSON rule descriptions files
- Documentation about how to integrate with CI/CD tools and exposes the HTML report
- Include new naming rules
- For external dependencies, allows to require a minimum version of the dependency.
- Ideas?

Thanks!

Any questions?



You can find more about Arch-Go at:

- <https://pkg.go.dev/github.com/fdaines/arch-go>
- <https://github.com/fdaines/arch-go>