Business logic without pain (Go edition)

Buzlov Ilya | 2020.08.09

Plan

- Introduction
- Problems
- Decision
- Questions

Services

180

People

200

Founded

2010

Modern

Docker, Golang, React, NodeJS

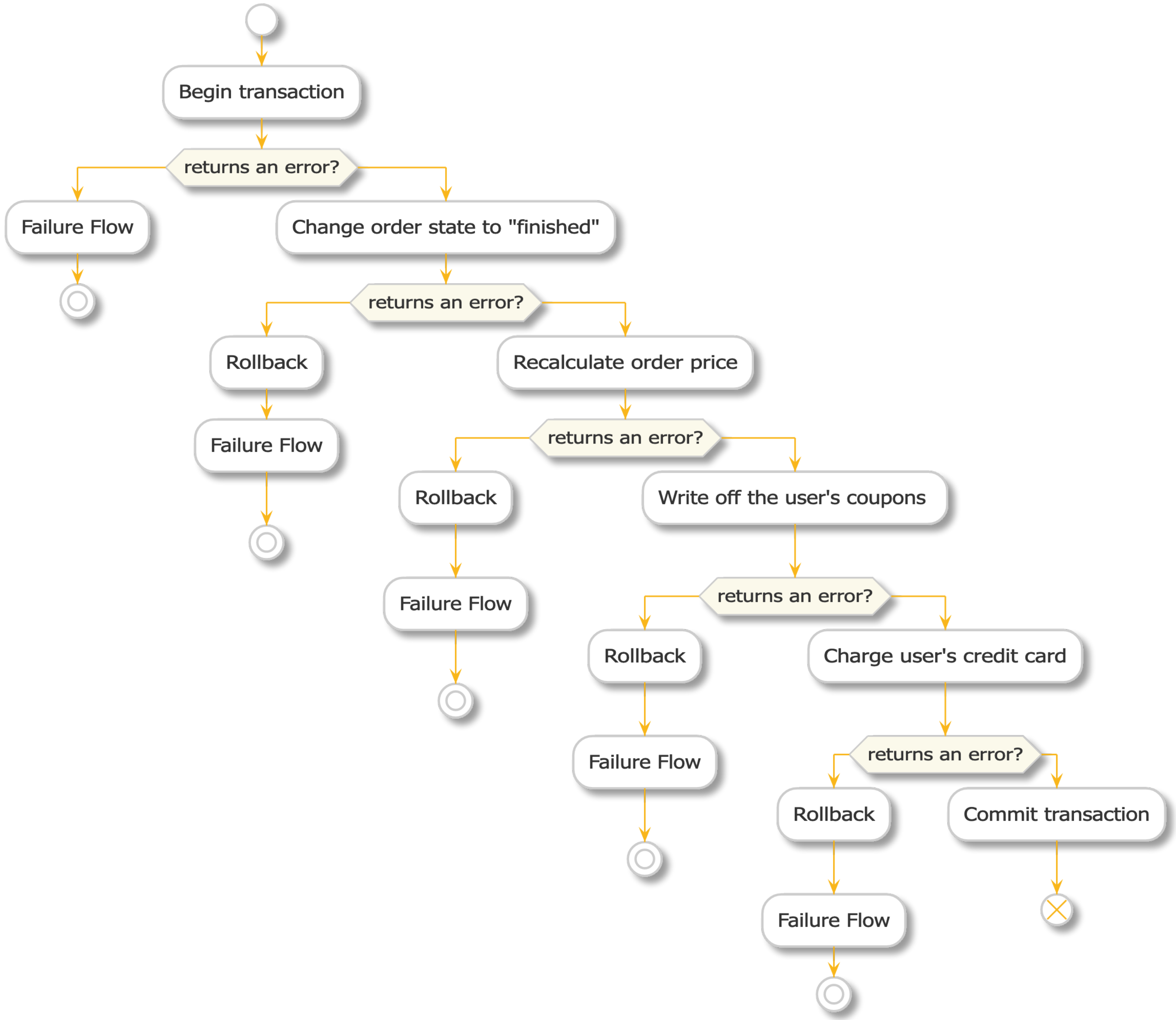
Countries

Russia, Israel, United Kingdom

Gett flexi

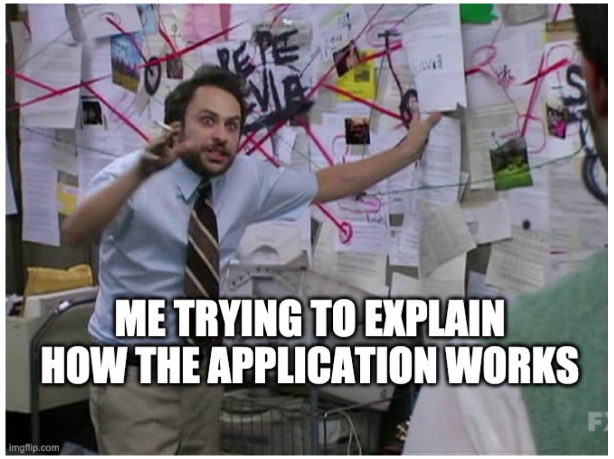
Big part of people work from home/remotely

Golang is the best language
Golang is the best language
Golang is the best language
Golang is the best language
Golang is the best language
Golang is the best language
Golang is the best language
Golang is the best language
MATT GROENING

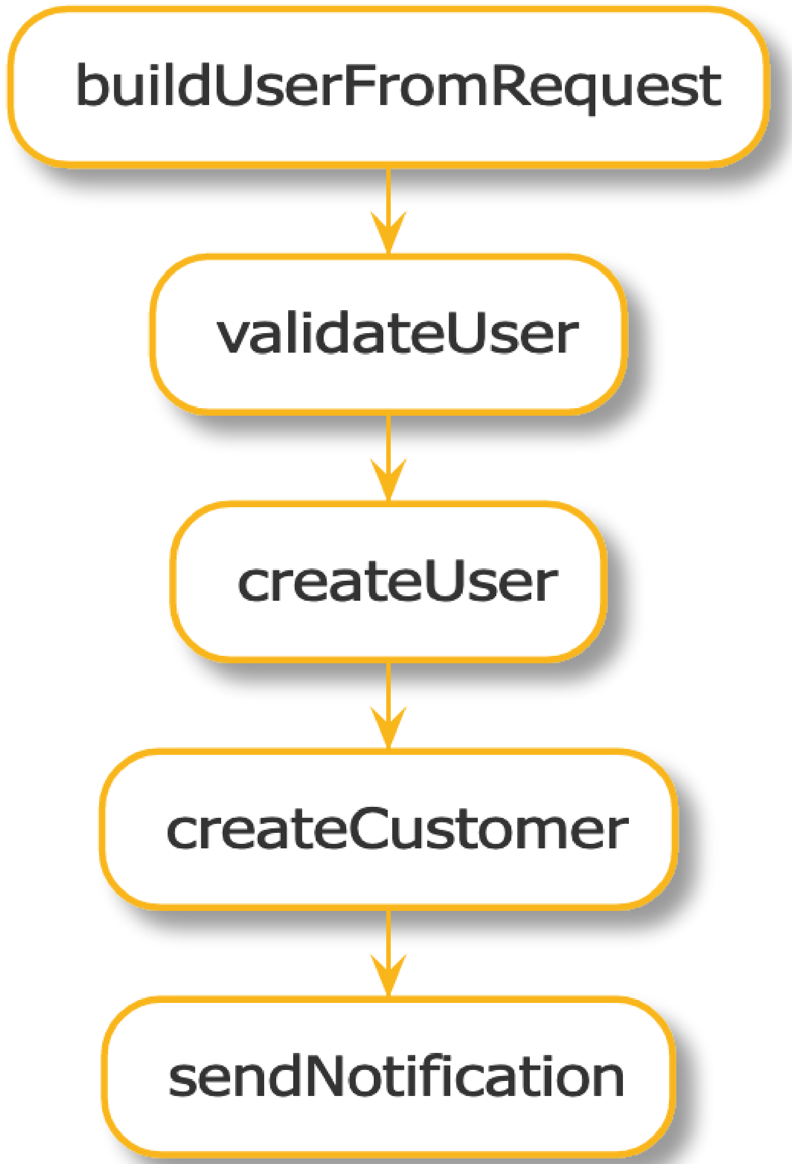

Begin transaction
returns an error?
Failure Flow
Change order state to "finished"
returns an error?
Rollback
Failure Flow
Recalculate order price
returns an error?
Rollback
Failure Flow
Write off the user's coupons
returns an error?
Rollback
Failure Flow
Charge user's credit card
returns an error?
Rollback
Failure Flow
Commit transaction

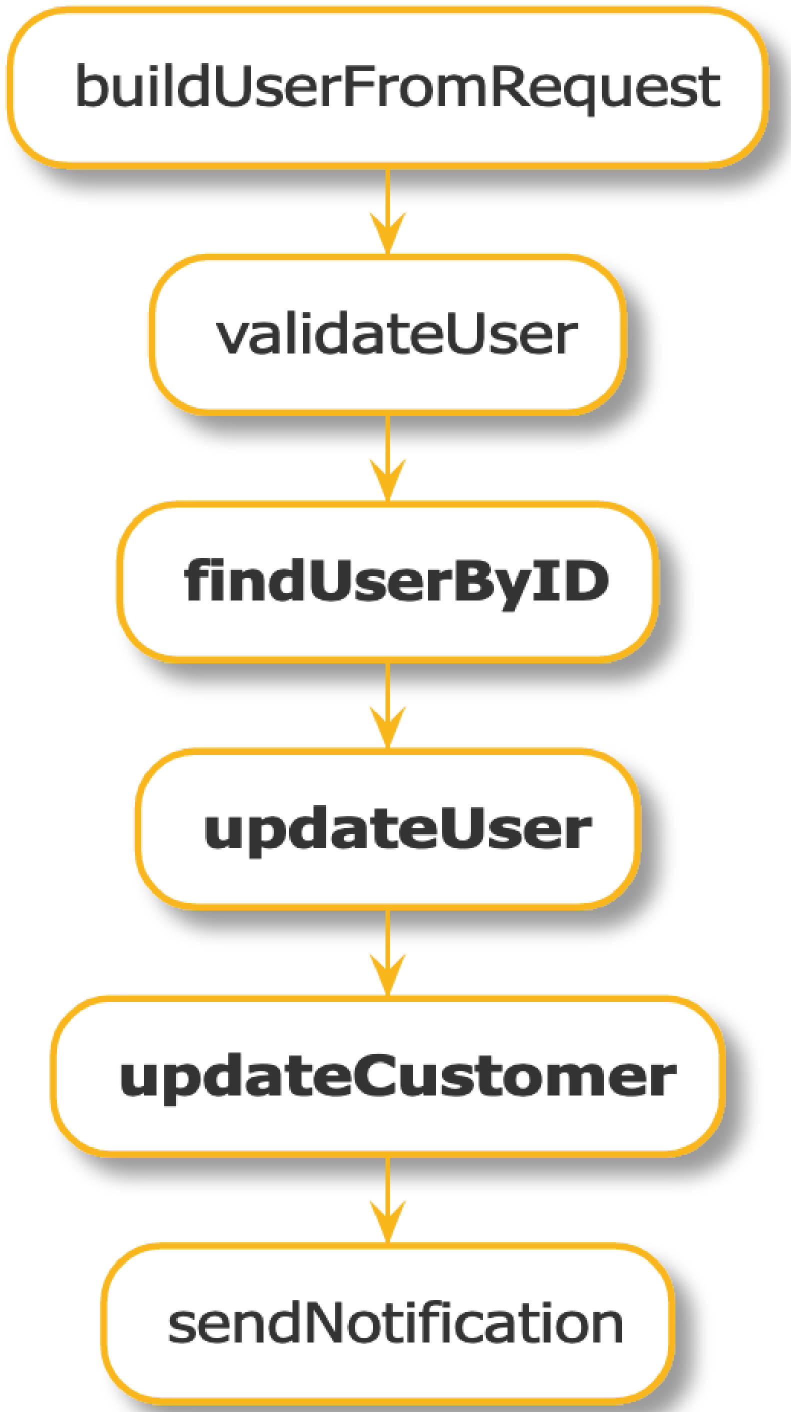
```go
func (o *OrderFinishedUseCase) Do(ctx context.Context, order Order) error {
    err := o.beginTransaction(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: beginTransaction")
    }
    err = o.changeStateToFinished(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: change state to finished")
    }
    err = o.recalculateOrderPrice(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: recalculate order price")
    }
    err = o.redeemUserCoupons(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: redeem user's coupons")
    }
    err = o.chargeUserCreditCard(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: charging a user")
    }
    err = o.commitTransaction(ctx, order)
    if err != nil {
        err = o.runFailureFlow(ctx, err, order)
        return errors.Wrap(err, "something goes wrong, step: commit transaction")
    }

    return nil
}
```

$$\begin{tikzcd}\bigcirc \arrow[r] & A \arrow[r] & B \arrow[r] & C \arrow[r] & D\end{tikzcd}$$

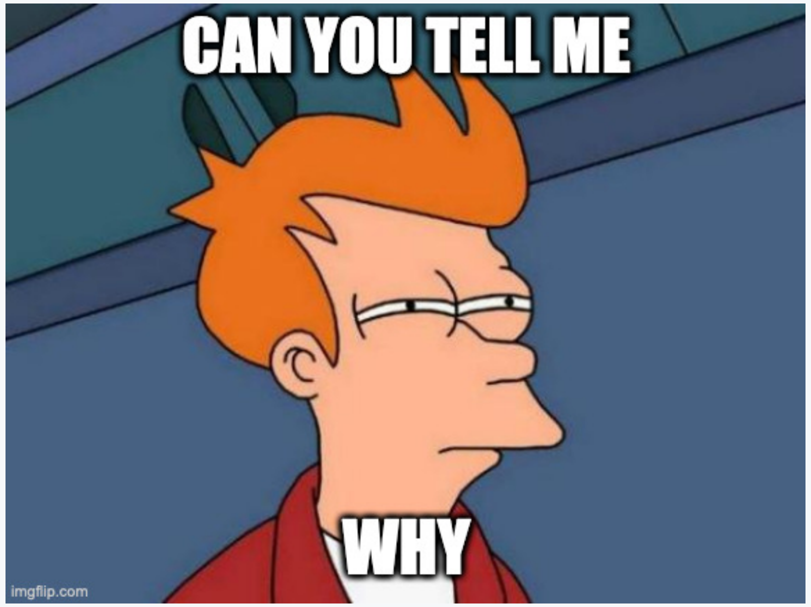


```go
var (
    xZoozRequestID = "123"
    entityID       = "123"
)
ctrl := gomock.NewController(t)
lockRepo := mocks.NewMockLockRepository(ctrl)
requestRepo := mocks.NewMockRequestRepository(ctrl)
errorRepo := mocks.NewMockErrorRepository(ctrl)
logger := mocks.NewMockLogger(ctrl)
systemSettingRepo := mocks.NewMockSystemSettingRepository(ctrl)
callbackRepo := mocks.NewMockCallbackWriter(ctrl)
systemSettingRepo.EXPECT().FindIntSystemSetting(ctx, "zooz_wait_time").Return(int64(1), nil)
lockRepo.EXPECT().Create(ctx, zoozRequestLock).Return(true, nil)
lockRepo.EXPECT().Delete(ctx, zoozRequestLock).Return(true, nil)
requestRepo.EXPECT().FindByTransactionID(ctx, entityID).Return(request, nil)
requestRepo.EXPECT().UpdateStateAndSetCompletedAt(ctx, request).Return(nil)
callbackRepo.EXPECT().Write(ctx, request).Return(nil)
lockRepo.EXPECT().Create(ctx, requestLock).Return(true, nil)
lockRepo.EXPECT().Delete(ctx, requestLock).Return(true, nil)
receiver := NewCallbackReceiver(lockRepo, requestRepo, errorRepo, logger, systemSettingRepo, callbackRepo)
```


buildUserFromRequest
validateUser
createUser
createCustomer
sendNotification


buildUserFromRequest
validateUser
findUserByID
updateUser
updateCustomer
sendNotification

```go
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
//...
if err != nil {
    return errors.Wrap(err, "something goes wrong")
}
```



POLICE
MOVIECLIPS.com

[adult swim]

Effe

- Provide visibility and traceability into these process flows
- Errors are wrapping automatically
- Dependencies build for flow automatically
- Easy flow debugging
- Easy flow extending
- Split dependencies for steps in flow: the step has only dependencies that it needs
- Allow greater reuse of existing functions
- Easy flow testing: small interface/easier to understanding what happening

```go
func buildUser() func(UserAttributes) User {
    return func(uAttrs UserAttributes) User {
        return User{
            Email:    uAttrs.Email,
            Password: uAttrs.Password,
        }
    }
}

func createUser(userRepo UserRepository) func(context.Context, User) error {
    return func(ctx context.Context, user User) error {
        return userRepo.Create(ctx, user)
    }
}
```

effe.go

```go
// +build effeinject

package actions

import (
    "github.com/GettEngineering/effe"
)

func BuildCreateUserFlow(uAttrs UserAttributes) error {
    effe.BuildFlow(
        effe.Step(buildUser),
        effe.Step(createUser),
    )
    return nil
}
```

```
effe
```

effe_gen.go

```go
// Code generated by Effe. DO NOT EDIT.

//+build !effeinject

package actions

import (
    "context"
)

type BuildCreateUserFlowFunc func(ctx context.Context, uAttrs UserAttributes)
type BuildCreateUserFlowService interface{
    //...
    //...
}
type BuildCreateUserFlowImpl struct {
    //...
    //...
}


func BuildCreateUserFlow(service BuildCreateUserFlowService) BuildCreateUserFlowFunc {
    return func(ctx context.Context, uAttrs UserAttributes) error {
        UserVal := service.BuildUser(uAttrs)
        err := service.CreateUser(ctx, UserVal)
        if err != nil {
            return errors.Wrap(err, "failed createUser")
        }
    return nil
    }
}
```

Service object

```go
type BuildCreateUserFlowService interface {
    BuildUser(uAttrs UserAttributes) User
    CreateUser(ctx context.Context, user User) error
}
```

Testing

```
u := User{}
serviceMock := mocks.NewBuildCreateUserFlow(ctrl)
createUserFlow := BuildCreateUserFlow(serviceMock)
serviceMock.EXPECT().BuildUser(uAttrs).Return(u)
serviceMock.EXPECT().CreateUser(u).Return(nil)
err := createUserFlow(ctx, zoozRequest)
assert.NoError(t, err)
```

Implementation

```go
type BuildCreateUserFlowImpl struct {
    buildUserFieldFunc  func(uAttrs UserAttributes) User
    createUserFieldFunc func(ctx context.Context, user User) error
}

func(b *BuildCreateUserFlowImpl) BuildUser(uAttrs UserAttributes) User {
    return b.buildUserFieldFunc(uAttrs)
}
```

Dependencies

```
func NewBuildCreateUserFlowImpl(userRepo UserRepository) *BuildCreateUserFlowImpl {
    return &BuildCreateUserFlowImpl{buildUserFieldFunc: buildUser(), createUserFieldFunc: createUser(userRepo)}
}
```

API

- Step
- Wrap
- Decision
- Failure

Wrap

```
func BuildCreateUserFlow(uAttrs UserAttributes) error {
    effe.BuildFlow(
        //...,
        effe.Wrap(effe.Before(beginTransaction), effe.Success(commitTransaction), effe.Failure(rollbackTransaction),
            effe.Step(createUser),
        )
        //...,
    )
    return nil
}
```

Generated code

```go
//...
err16 := func(ctx context.Context, user User) error {
  transaction err15 := service.BeginTransaction(ctx)
  if err15 != nil {
    err15 = service.RollbackTransaction(ctx, err15, transaction)
    return errors.Wrap(err15, "failed beginTransaction")
  }

  err14 := service.CreateUser(ctx, user)
  if err != nil {
    err14 = service.RollbackTransaction(ctx, err15, transaction)
    return errors.Wrap(err14, "failed createUser")
  }

  err13 := service.CommitTransaction(transaction)
  if err != nil {
    err13 = service.RollbackTransaction(ctx, err13, transaction)
    return  errors.Wrap(err13, "failed commitTransaction")
  }
  return nil
}
//...
```

Decision

```
func BuildCreateUserFlow(uAttrs UserAttributes) error {
    effe.BuildFlow(
        //....,
            effe.Decision(new(entities.LockCreated),
                effe.Case(false, effe.Step(stop())),
                effe.Case(true,
                    effe.Step(createUser),
                    effe.Step(createCustomer),
                ),
        //....,
    )
    return nil
)
```

Generated code

```go
//...
err15 = func(lockCreatedVal entities.LockCreated, user User) error {
   switch lockCreatedVal {
      case true:
        err := func(ctx context.Context) {
          err = createUser(user)
          if err != nil {
            return errors.Wrap("failed createUser")
          }
          err = createCustomer(user)
          if err != nil {
            return errors.Wrap("failed createUser")
          }
        }()
      case false:
        service.Stop()
        return nil
      default:
        return errors.New("unsupported type lockCreatedVal")
   }
}
//...
```

Failure

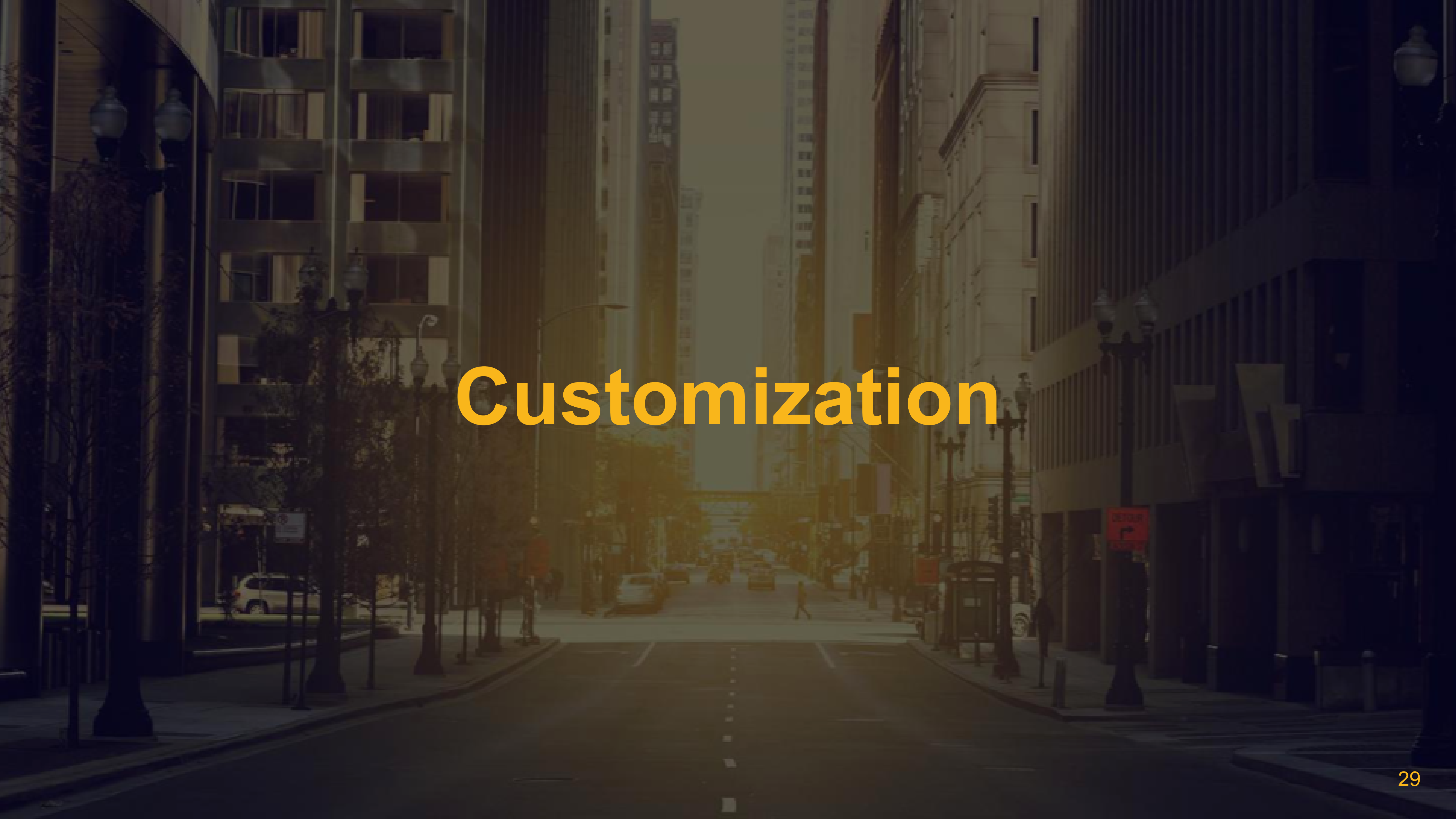
- BuildFlow
- Wrap
- Decision

Flow

```
func BuildCreateUserFlow(uAttrs UserAttributes) error {
    effe.BuildFlow(
        effe.Step(buildUser),
        effe.Step(createUser),
        effe.Failure(handleErr),
    )
    return nil
)
```

Generated

```
func BuildCreateUserFlow(service BuildCreateUserFlowService) BuildCreateUserFlowFunc {
    return func( ctx context.Context, uAttrs UserAttributes) error {
        UserVal := service.BuildUser(uAttrs)
        err := service.CreateUser(ctx, UserVal)
        if err != nil {
            err = service.HandleErr(err)
            return errors.Wrap(err, "failed createUser")
        }
    return nil
    }
}
```



Customization

```go
func BuildMyFlow() error {
    effe.BuildFlow(
        effe.Step(step1),
        mygenerator.POST(
            "http://example.com",
        ),
    )
    return nil
}
```

DSL

```go
package mygenerator

import (
  "github.com/GettEngineering/effe"
)

func POST(url string) interface{} {
    panic("implementation is not generated, run myeffe")
}

func LoadPostRequestComponent(effeConditionCall *ast.CallExpr, f loaders.FlowLoader) (types.Component, error) {
    return &PostRequestComponent{
        URI: effeConditionCall.Args[0],
    }, nil
}

func GenPostRequestComponent(f strategies.FlowGen, c types.Component) (strategies.ComponentCall, error) {
    component, ok := c.(*PostRequestComponent)
    if !ok {
        return nil, errors.Errorf("component %s is not a component with type PostRequestComponent", component.Name())
    }
    return &postComponentCall{
        input:  &ast.FieldList{},
         output: output,
        fn:     fn,
    }
}
```

Generator

```
    settings := generator.DefaultSettigs()
    strategy := strategies.NewChain(strategies.WithServiceObjectName(settings.LocalInterfaceVarname()))
    err := strategy.Register("POST", GenPostRequestComponent)
    require.NoError(t, err)

    loader := loaders.NewLoader(loaders.WithPackages([]string{"effe", "testcustomization"}))
    err = loader.Register("POST", LoadPostRequestComponent)
    require.NoError(t, err)

    gen := generator.NewGenerator(
        generator.WithSetttings(settings),
        generator.WithLoader(loader),
        generator.WithStrategy(strategy),
    )
```

```go
func C(service CService) CFunc {
	return func() (*gentleman.Response, error) {
		err := service.Step1()
		if err != nil {
			return nil, err
		}
		responsePtrVal, err := func() (*gentleman.Response, error) {
			cli := gentleman.New()
			cli.URI("http://example.com")
			req := cli.Request()
			req.Method(POST)
			return cli.Send()
		}()
		if err != nil {
			return responsePtrVal, err
		}
		return responsePtrVal, nil
	}
}
```

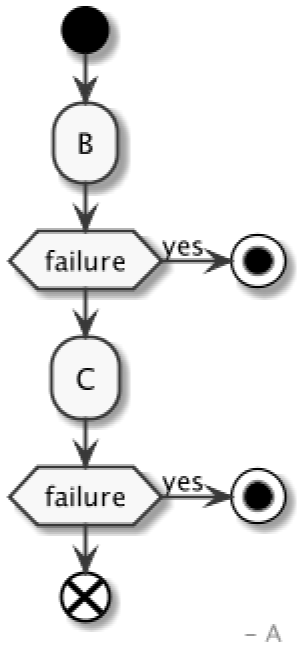
Reusing

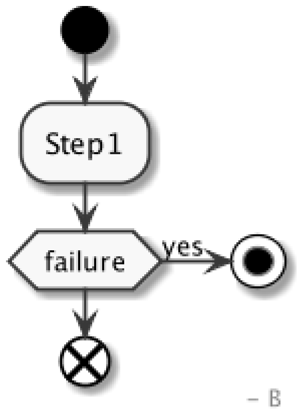
```go
func A() error {
    effe.BuildFlow(
        effe.Step(B),
        effe.Step(C),
    )
    return nil
}
func B() error {
    effe.BuildFlow(
        effe.Step(step1),
    )
    return nil
}
func C() error {
    effe.BuildFlow(
        effe.Step(B),
    )
    return nil
}
```

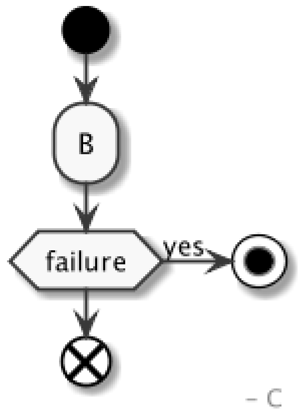
Diagrams

```
//go:generate go run ../../cmd/effe/main.go
//go:generate go run ../../cmd/effe/main.go -d -out ./doc/
```

```
$ effe -d -out ./doc
effe: wrote foo/doc/B.plantuml
effe: wrote foo/doc/C.plantuml
effe: wrote A.plantuml
```


B
failure
yes
C
failure
yes
– A


Step1
failure
yes
– B


B
failure
yes
– C

https://github.com/GettEngineering/effe/