Discovering Better **Object Oriented Design** with **Test**

Discovering ~~Better~~ SEXY Object Oriented Design with Test

SEXY

Discovering ~~Better~~ Object Oriented Design with Test

Github/Twitter/Weibo

@poiyzy

Why hate to write test?

Why hate to write test?

* Protect our code.

Why hate to write test?

* Protect our code.

* Hard to think about how to write it.

Why hate to write test?

* Protect our code.

* Hard to think about how to write it.

* It is painful, can’t keep on going.

Why hate to write test?

* Protect our code.

Fetch up test after implementation!

* Hard to think about how to write it.

* It is painful, can’t keep on going.

Why hate to write test?

It is not TDD.

~~Fetch up test after implementation!~~

* Protect our code.

* Hard to think about how to write it.

* It is painful, can’t keep on going.

Something behind that

* Hard to think about how to write it.

* It is painful, can't keep on going.

Something behind that

* Hard to think about how to write it.

* Write it Step by Step, test reflect your design structure.

* It is painful, can't keep on going.

Something behind that

* Hard to think about how to write it.

* Write it Step by Step, test reflect your design structure.

* It is painful, can’t keep on going.

* It is a signal about your bad design.

Why need OO design?

Why need OO design?

* Code is unclearly.

Why need OO design?

* Code is unclearly.

* Hard to understand.

Why need OO design?

* Code is unclearly.

* Hard to understand.

* It is unmaintainable.

Refactor the bad structure as earlier as possible.

Refactor the bad structure as earlier as possible.

Test reflects the design structure.

Refactor the bad structure as earlier as possible.

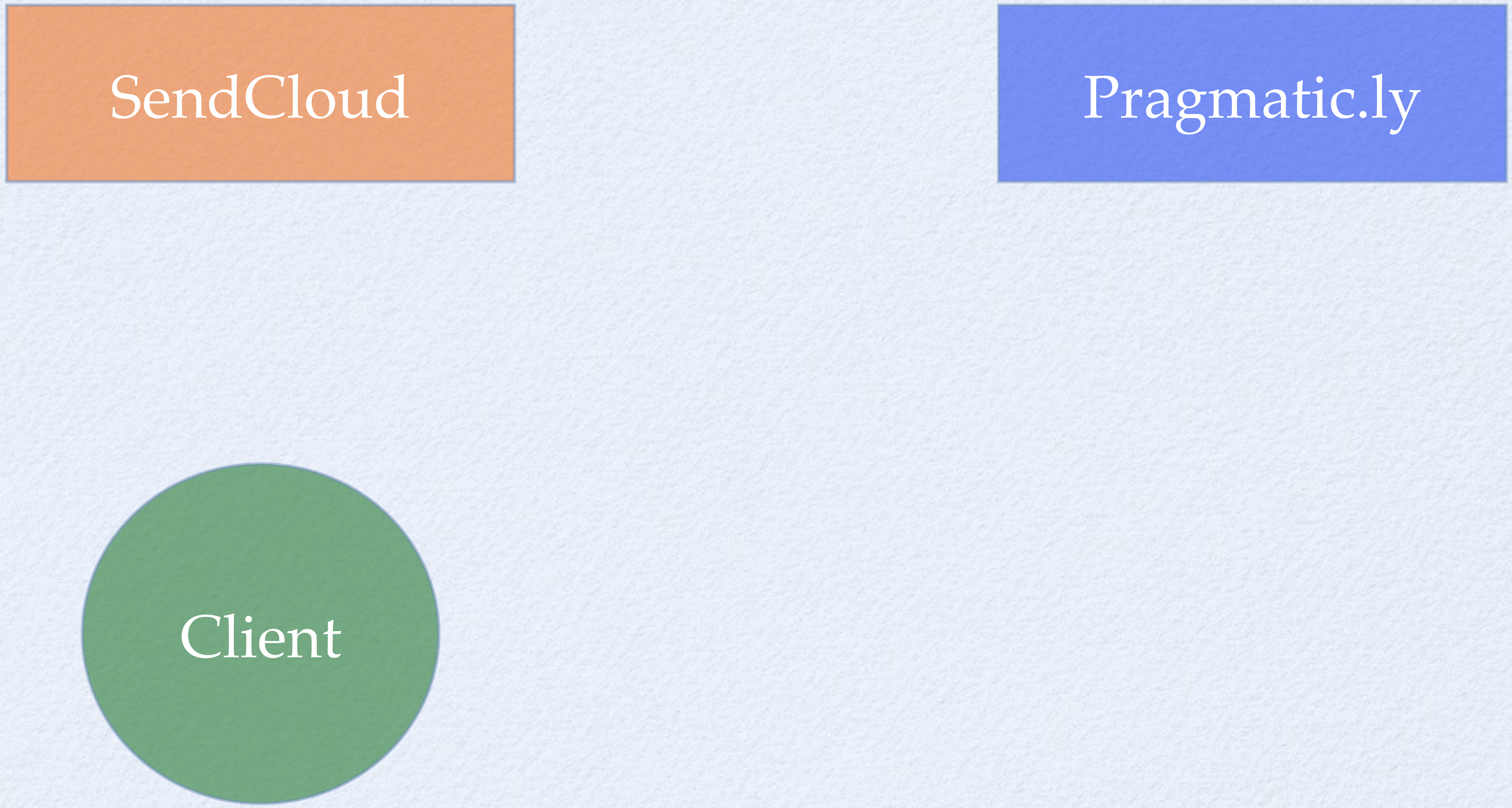
Test reflects the design structure.

Discovering better OO design with test

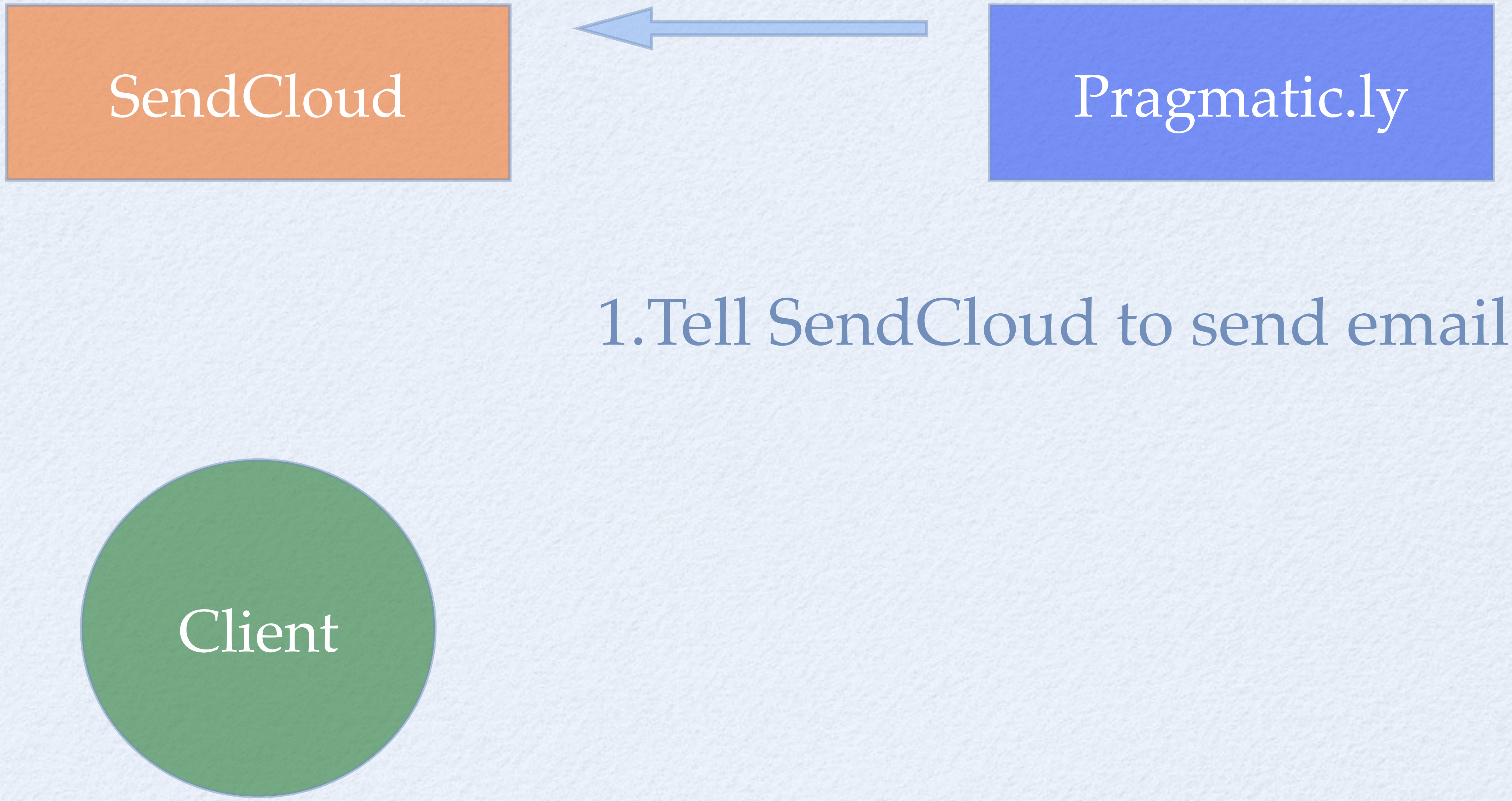
A New Feature

of Pragmatic.ly

Working Flow

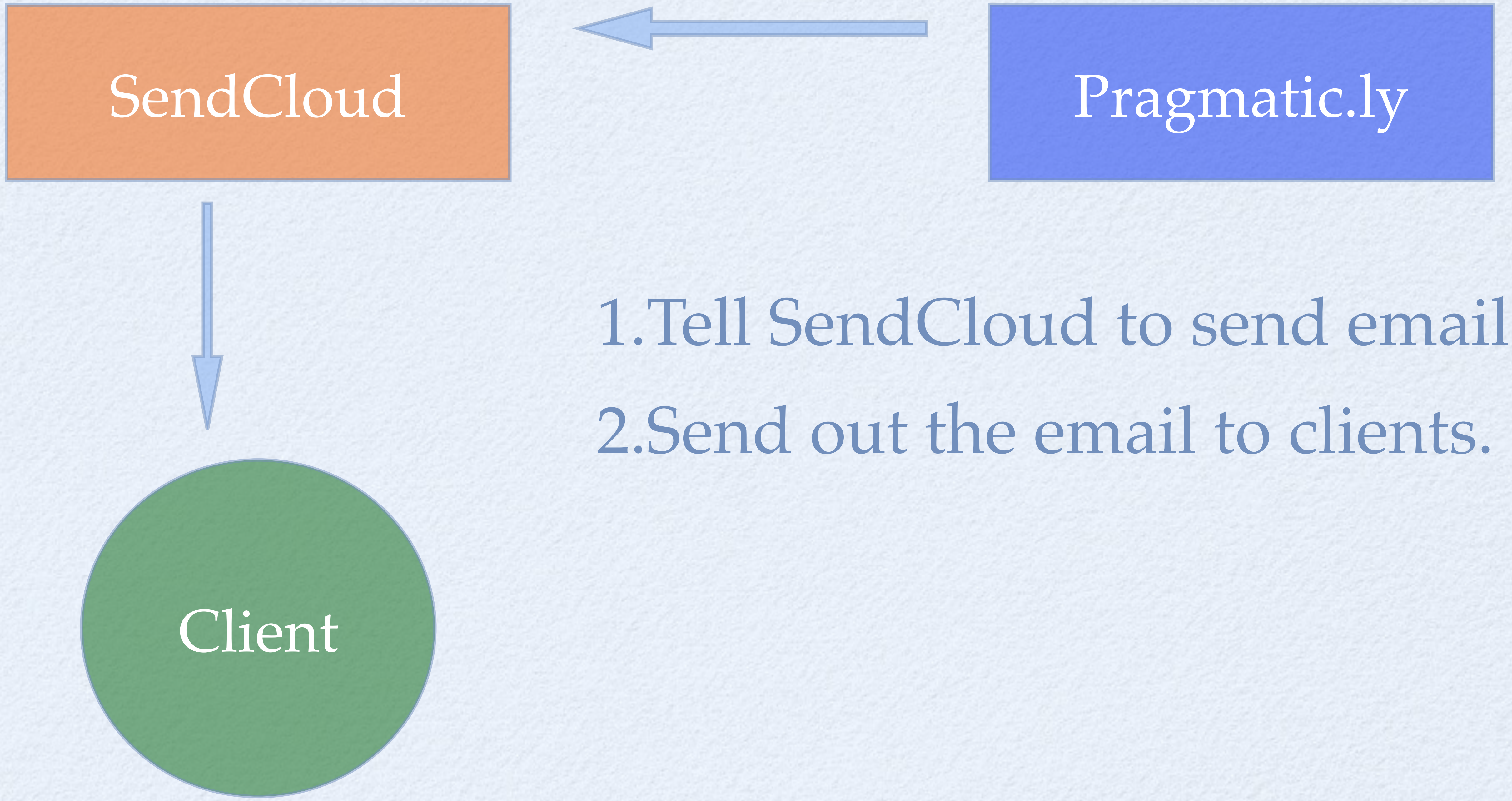


Working Flow



1.Tell SendCloud to send email.

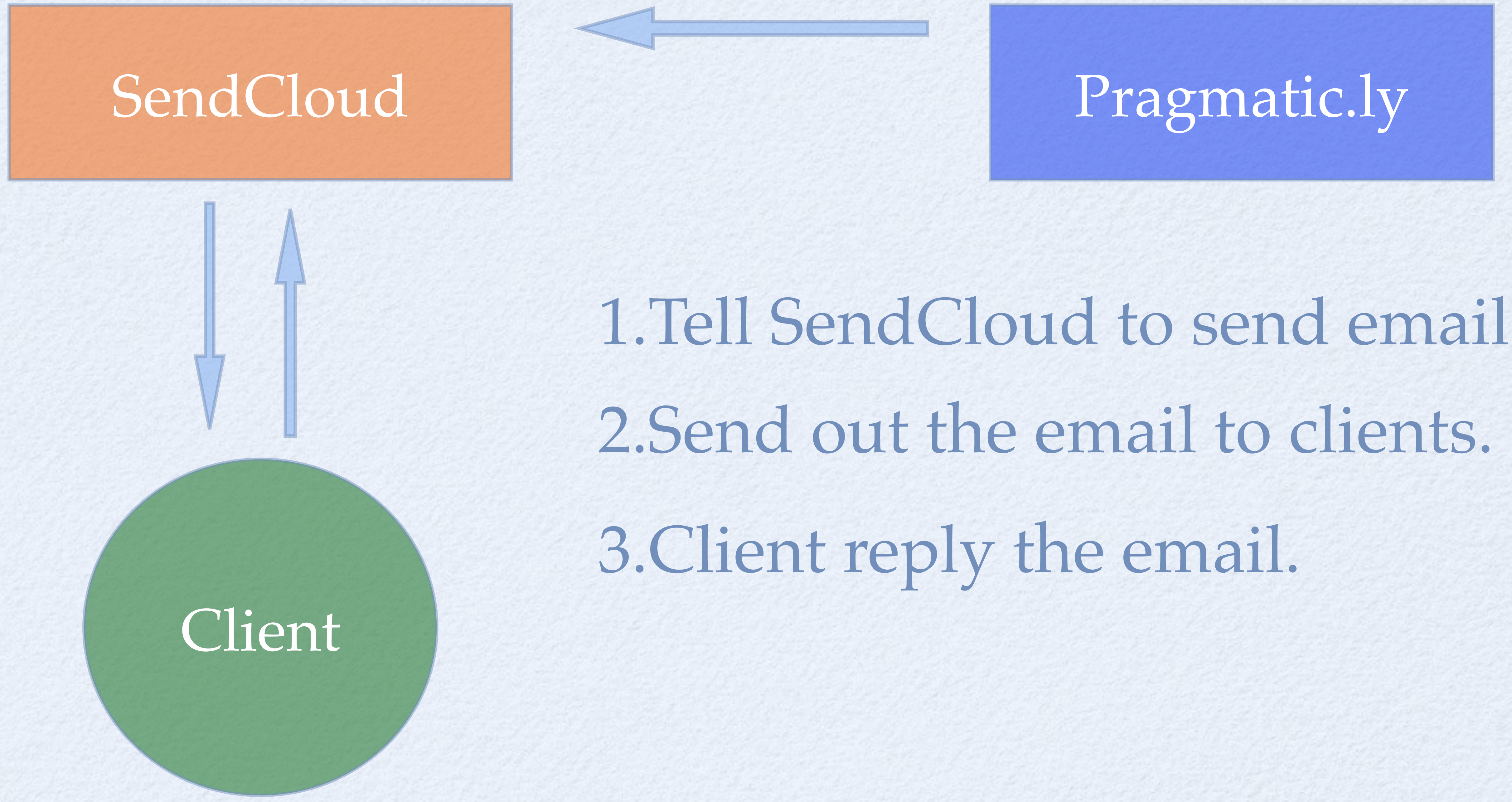
Working Flow



1.Tell SendCloud to send email.

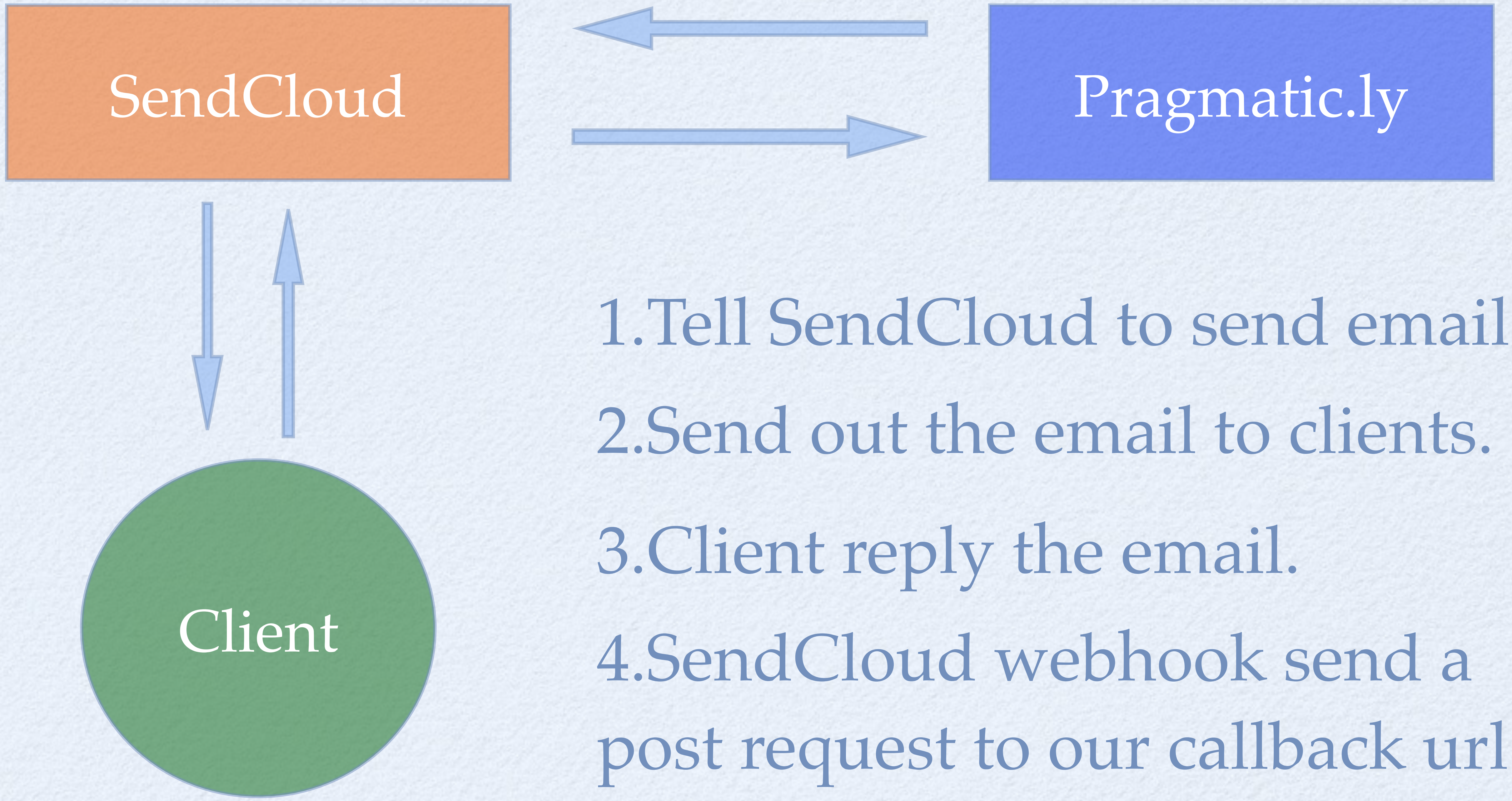
2.Send out the email to clients.

Working Flow



1. Tell SendCloud to send email.
2. Send out the email to clients.
3. Client reply the email.

Working Flow



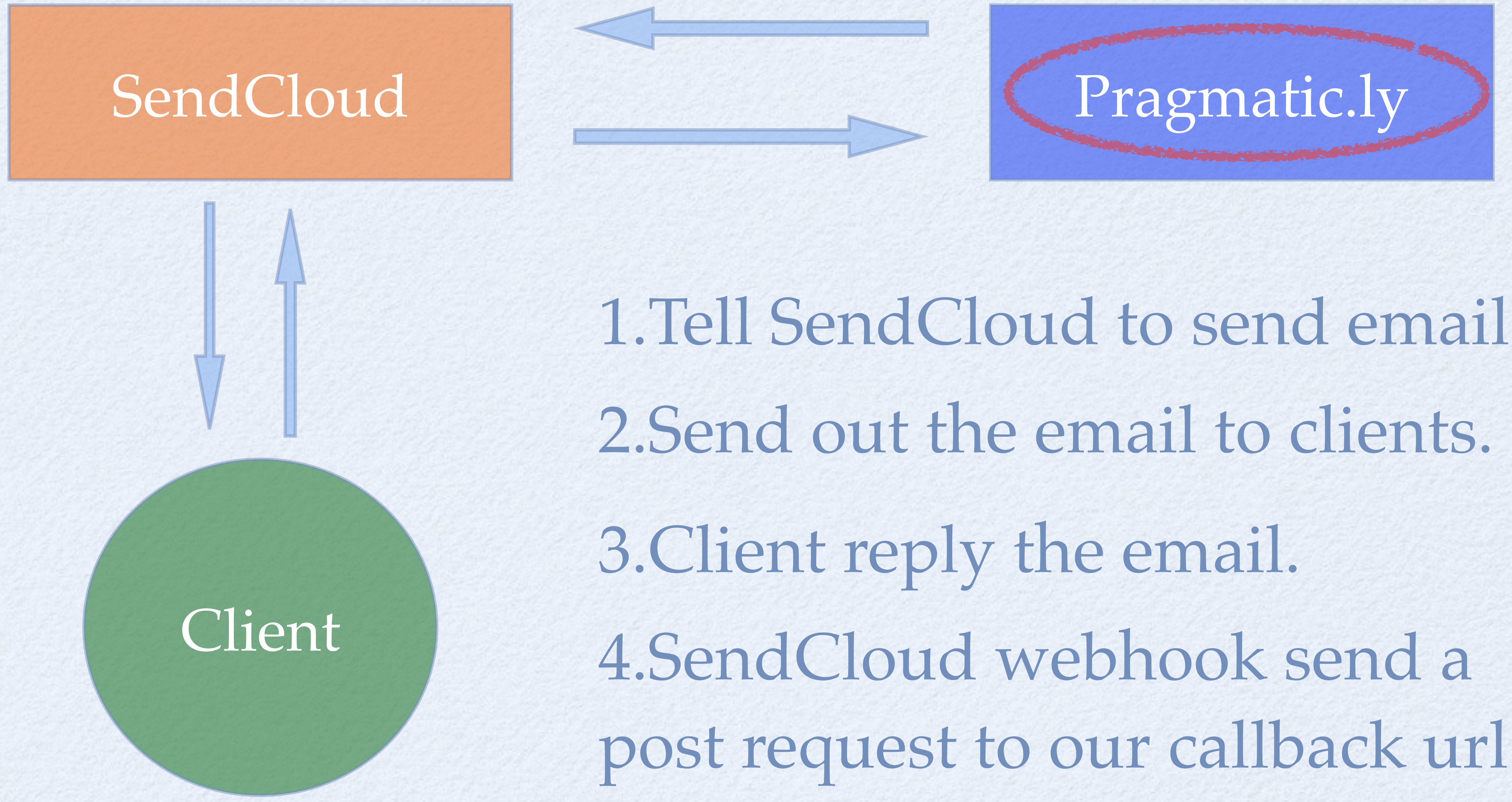
1.Tell SendCloud to send email.

2.Send out the email to clients.

3.Client reply the email.

4.SendCloud webhook send a post request to our callback url.

Working Flow



1.Tell SendCloud to send email.

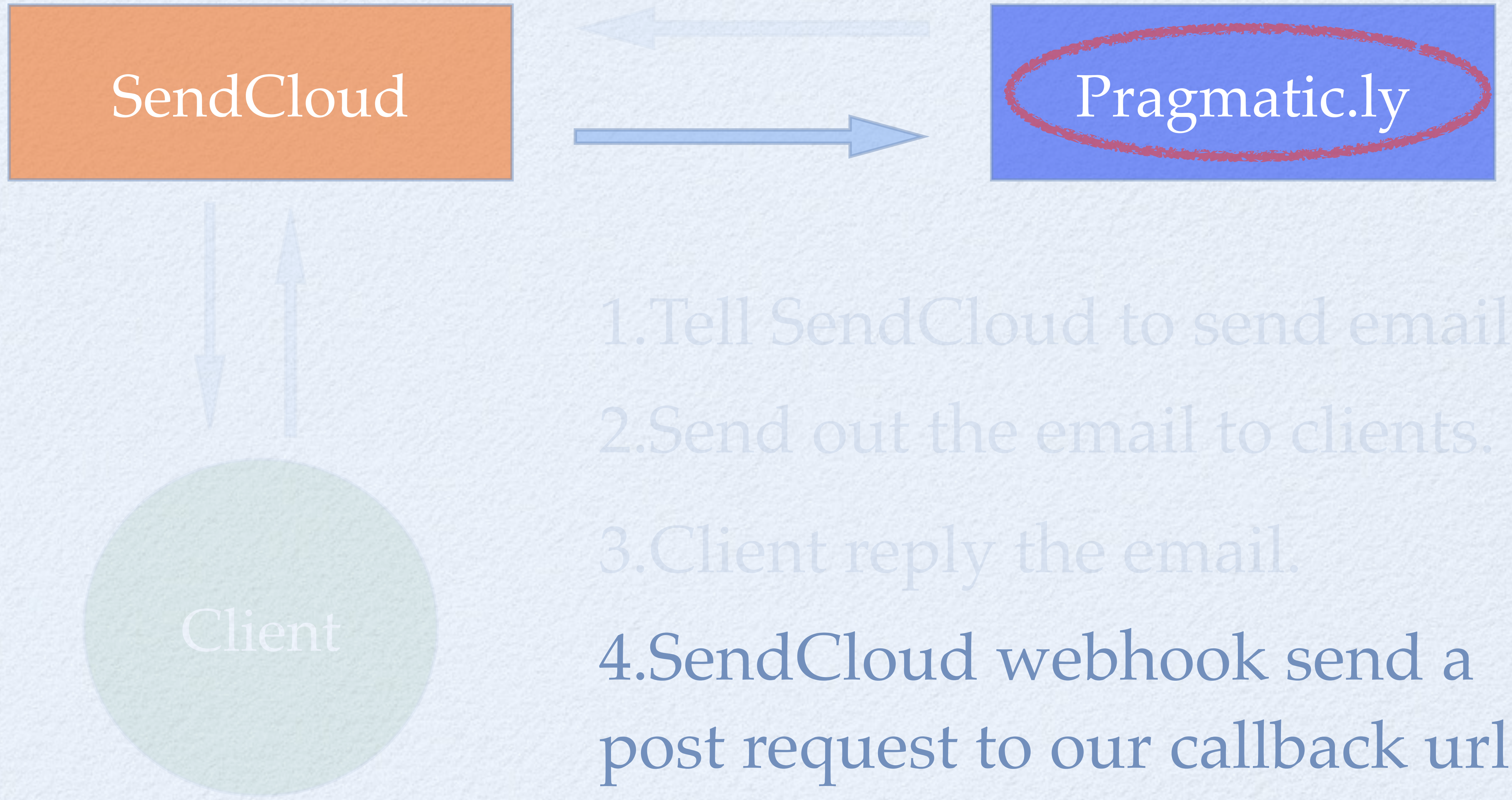
2.Send out the email to clients.

3.Client reply the email.

4.SendCloud webhook send a post request to our callback url.

5.Create a comment.

Working Flow



1.Tell SendCloud to send email.

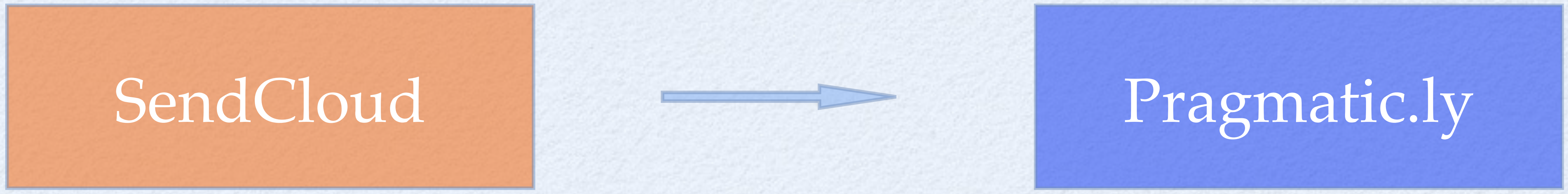
2.Send out the email to clients.

3.Client reply the email.

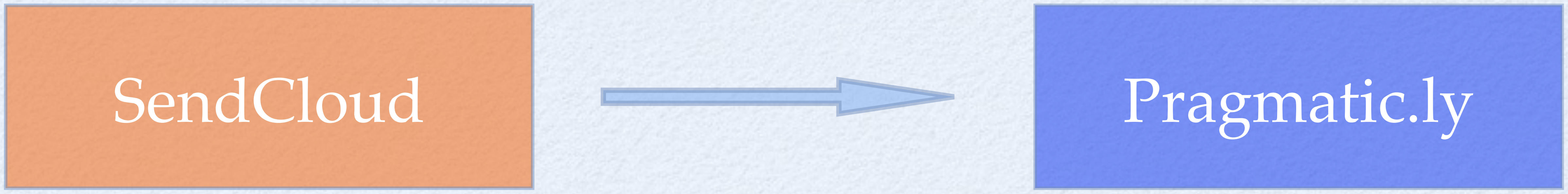
4.SendCloud webhook send a post request to our callback url.

5.Create a comment.

What we need to do?

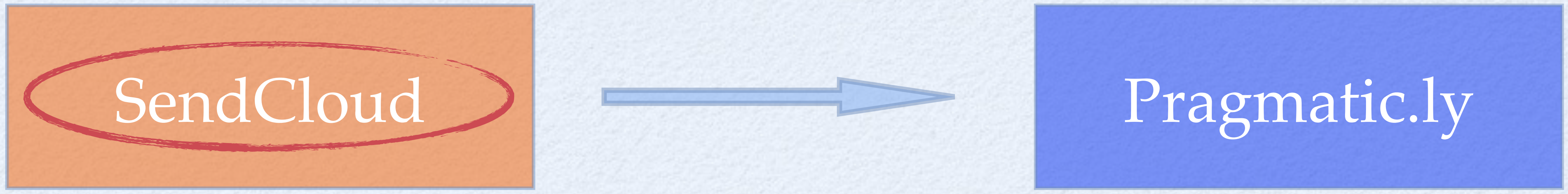


What we need to do?



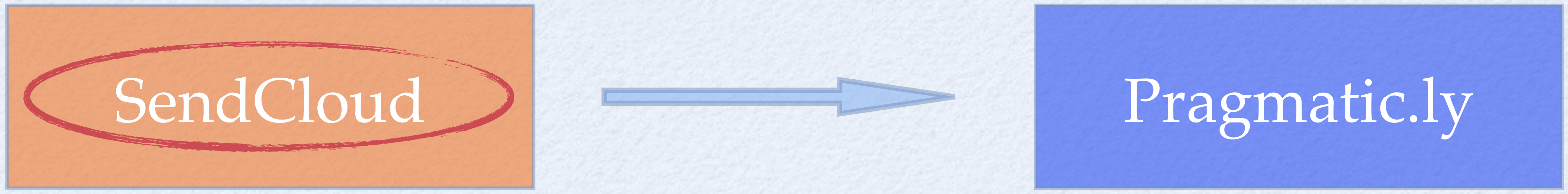
1. Verify the post request.

What we need to do?



1. Verify the post request.

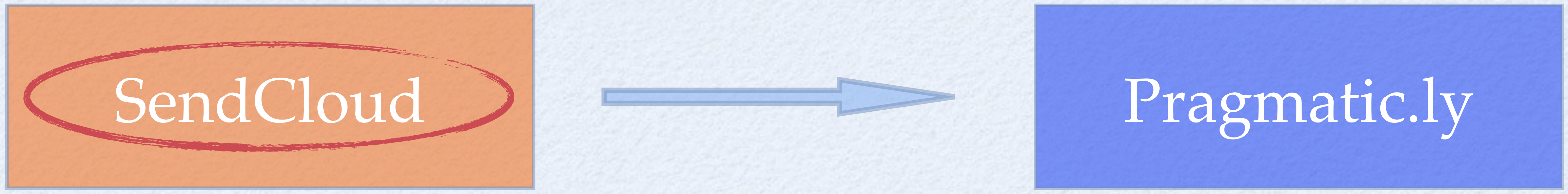
What we need to do?



1. Verify the post request.

Reply-To: ticket+PROJECT_UID+TICKET_UID@info.pragmatic.ly

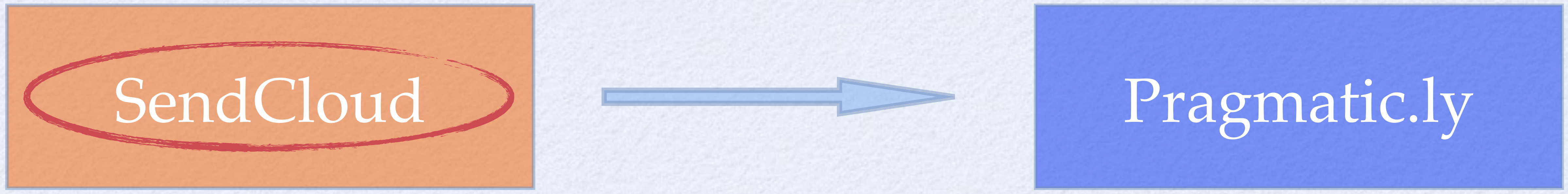
What we need to do?



1. Verify the post request.
2. Validate the reply email information.

Reply-To: ticket+PROJECT_UID+TICKET_UID@info.pragmatic.ly

What we need to do?



1. Verify the post request.
2. Validate the reply email information.

   Reply-To: ticket+PROJECT_UID+TICKET_UID@info.pragmatic.ly

3.Create a comment.

Implementation

TDD

Where to begin with

Where to begin with

I need a callback url

Where to begin with

I need a callback url

```
class EmailRepliesController < ApplicationController
  def create
  end
end
```

Where to begin with

I need a callback url

```
class EmailRepliesController < ApplicationController
  def create
  end
end
```

Test First

Where to begin with

I need a callback url

```
class EmailRepliesController < ApplicationController
  def create
  end
end
```

Test First

```
describe EmailRepiesController do
  describe "POST create" do
  end
end
```

Verify Post Request Source

```ruby
describe EmailRepiesController do
  describe "POST create" do







  end
end
```

Verify Post Request Source

```ruby
describe EmailRepiesController do
  describe "POST create" do
    context "when the post request is a valid request" do

    end




  end
end
```

Verify Post Request Source

```
describe EmailRepiesController do
  describe "POST create" do
    context "when the post request is a valid request" do


    end

    context "when the post request is not a valid request" do

    end
  end
end
```

Verify Post Request Source

```
describe EmailRepiesController do
  describe "POST create" do
    context "when the post request is a valid request" do
      it "returns status 200"
    end

    context "when the post request is not a valid request" do

    end
  end
end
```

Verify Post Request Source

```
describe EmailRepiesController do
  describe "POST create" do
    context "when the post request is a valid request" do
      it "returns status 200"
    end

    context "when the post request is not a valid request" do
      it "returns status 422"
    end
  end
end
```

What we need to do?

$$\begin{tikzcd}\text{SendCloud} \arrow[r] & \text{Pragmatic.ly}\end{tikzcd}$$

1. Verify the post request.
2. The reply email information.

   Reply-To: ticket+PROJECT_UID+TICKET_UID@info.pragmatic.ly

3.Create a comment.

What we need to do?

$$\begin{tikzcd}\text{SendCloud} \arrow[r] & \text{Pragmatic.ly}\end{tikzcd}$$

~~1. Verify the post request.~~

2. The reply email information.

   Reply-To: ticket+PROJECT_UID+TICKET_UID@info.pragmatic.ly

3.Create a comment.

Validation & Creation

```
describe "POST create" do
  context "when the post request is a valid request" do
    it "returns status 200"








  end
end
```

Validation & Creation

```
describe "POST create" do
  context "when the post request is a valid request" do
    it "returns status 200"






  end
end
```

Validation & Creation

```
describe "POST create" do
  context "when the post request is a valid request" do
    it "returns status 200"

    context "when the project uid is valid" do
    end





  end
end
```

Validation & Creation

```
describe "POST create" do
  context "when the post request is a valid request" do
    it "returns status 200"

    context "when the project uid is valid" do
    end


    context "when the project uid is invalid" do

    end
  end
end
```

Validation & Creation

```
describe "POST create" do
  context "when the post request is a valid request" do
    it "returns status 200"

    context "when the project uid is valid" do
    end


    context "when the project uid is invalid" do
      it "doesn't create the comment"
    end
  end
end
```

Validation & Creation

```
context "when the project uid is valid" do









end
```

Validation & Creation

```
context "when the project uid is valid" do
  context "when the user email is valid" do










  end



end
```

Validation & Creation

```
context "when the project uid is valid" do
  context "when the user email is valid" do









  end
  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end
```

Validation & Creation

```
context "when the project uid is valid" do
  context "when the user email is valid" do
    context "when the user has the right to access this
    project" do



    end





  end
  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end
```

Validation & Creation

```
context "when the project uid is valid" do
  context "when the user email is valid" do
    context "when the user has the right to access this
    project" do



    end
    context "when the user has no right to access this
    project" do
      it "doesn't create the comment"
    end
  end
  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end
```

Validation & Creation

```
context "when the project uid is valid" do
  context "when the user email is valid" do
    context "when the user has the right to access this
    project" do
      # Next Step
    end
    context "when the user has no right to access this
    project" do
      it "doesn't create the comment"
    end
  end
  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end
```

Validation & Creation

```
context "when the user has the right to access this project" do



end
```

Validation & Creation

```
context "when the user has the right to access this project" do
  context "when the ticket uid is valid" do


  end



end
```

Validation & Creation

```
context "when the user has the right to access this project" do
  context "when the ticket uid is valid" do


  end
  context "when the ticket uid is invalid" do

  end
end
```

Validation & Creation

```
context "when the user has the right to access this project" do
  context "when the ticket uid is valid" do
    it "creates the comment"
  end
  context "when the ticket uid is invalid" do

  end
end
```

Validation & Creation

```
context "when the user has the right to access this project" do
  context "when the ticket uid is valid" do
    it "creates the comment"
  end
  context "when the ticket uid is invalid" do
    it "doesn't create the comment"
  end
end
```

What do we get now?

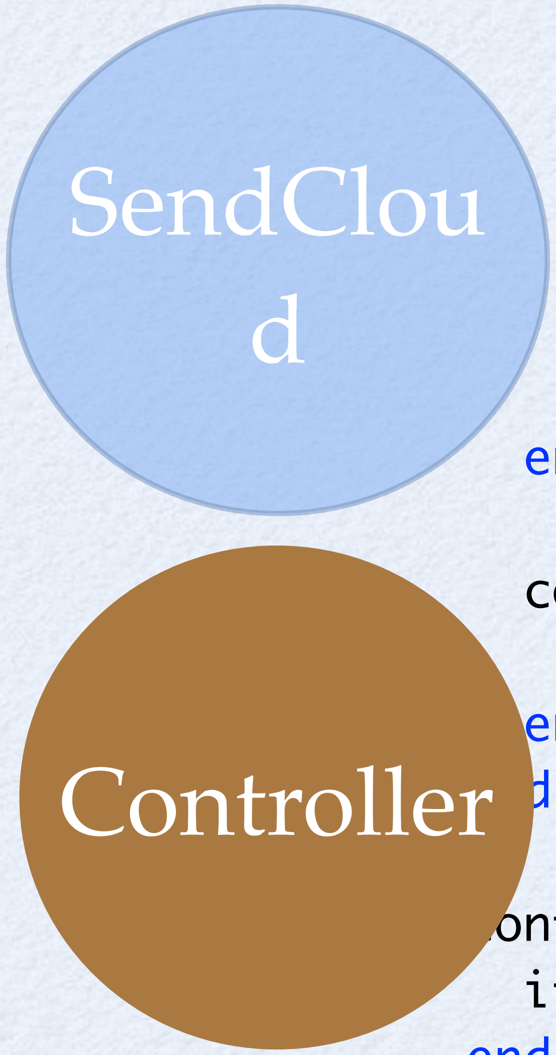
```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

What do we get now?

```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
    d

   ontext "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

Controller

What do we get now?



```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
     d

   ontext "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

What do we get now?

```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
    d

   ontext "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

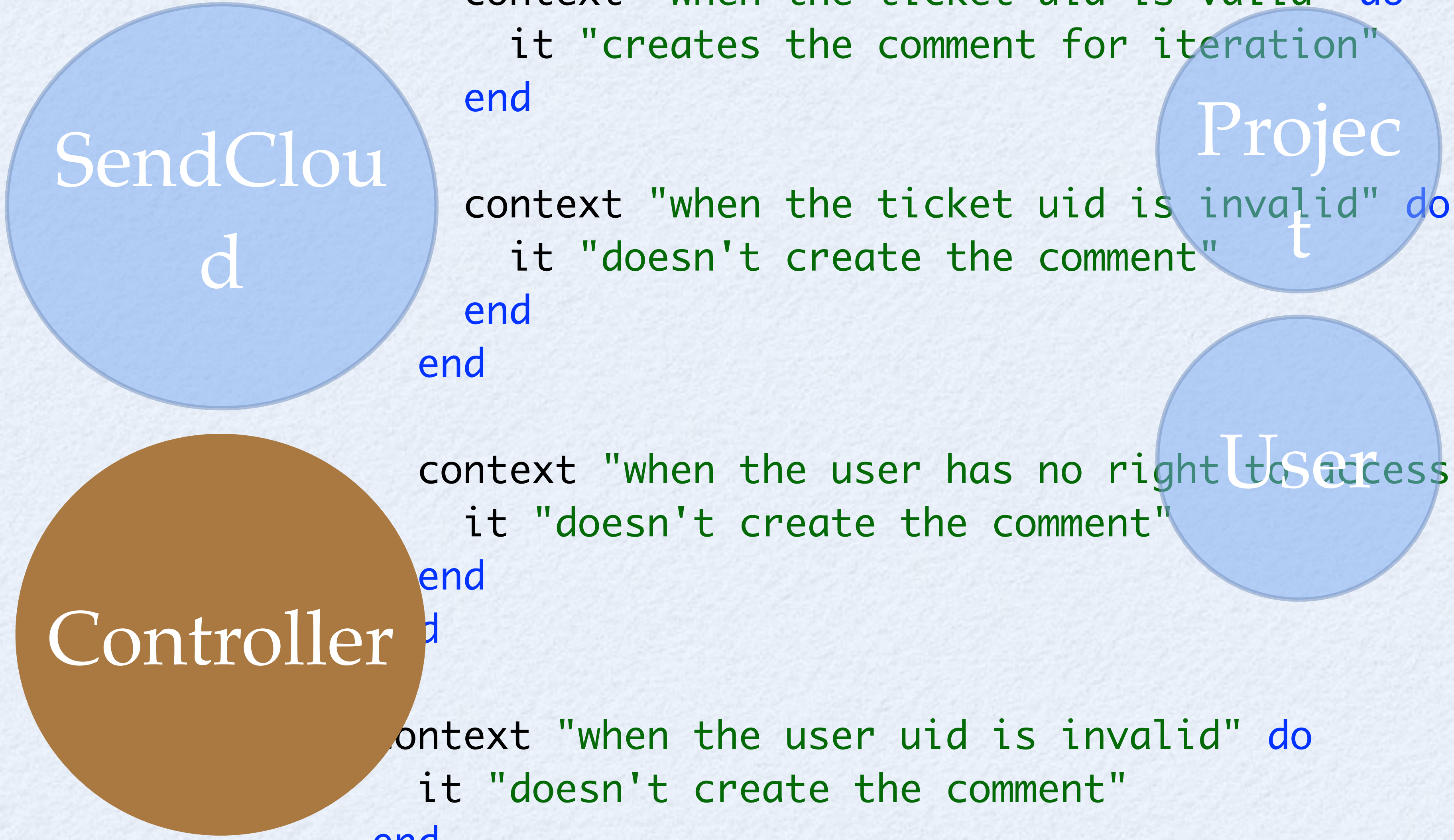
SendClou
d

Projec
t

Controller

What do we get now?

```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```



What do we get now?

```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

Ticket

SendCloud

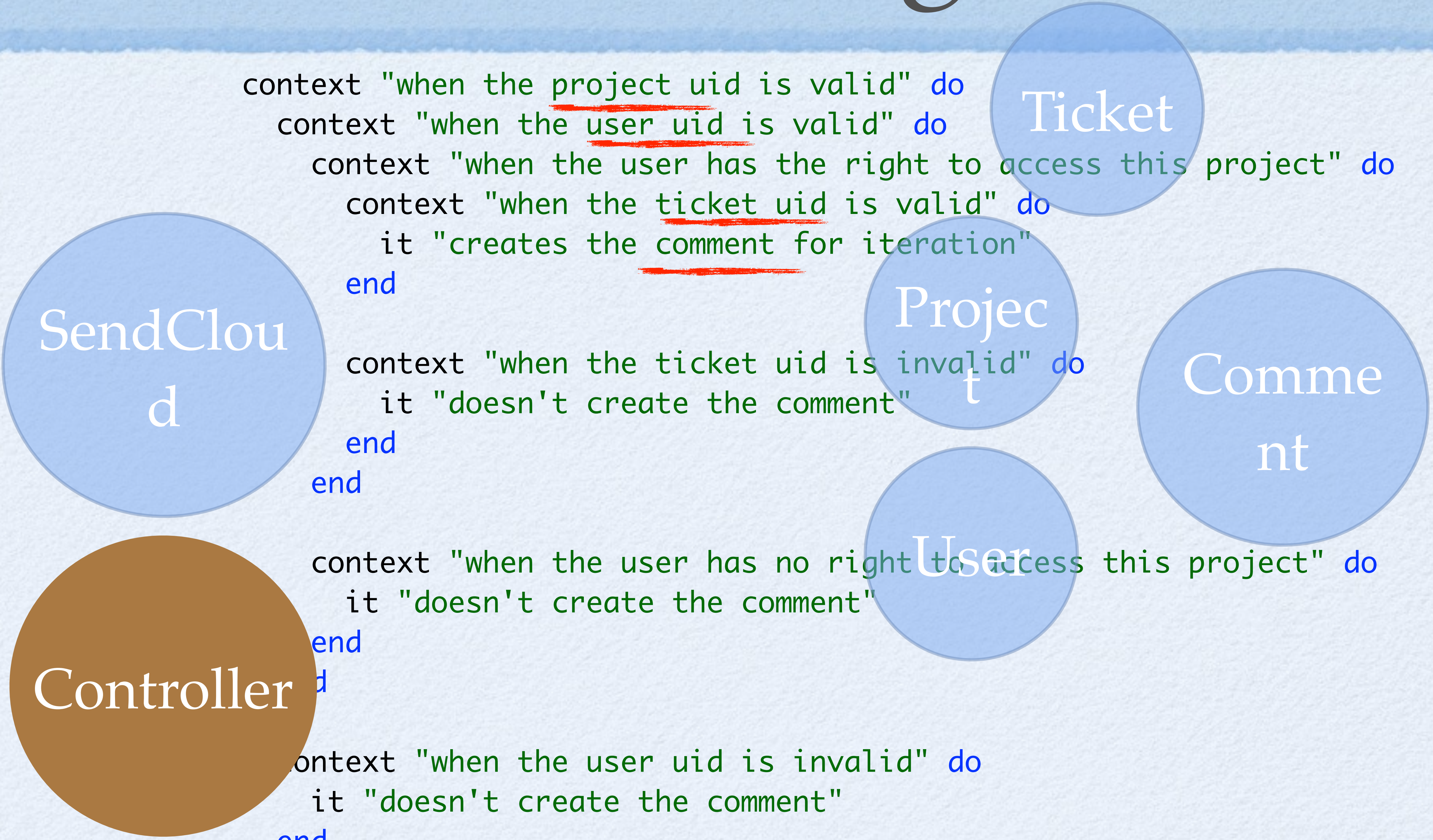
Project

User

Controller

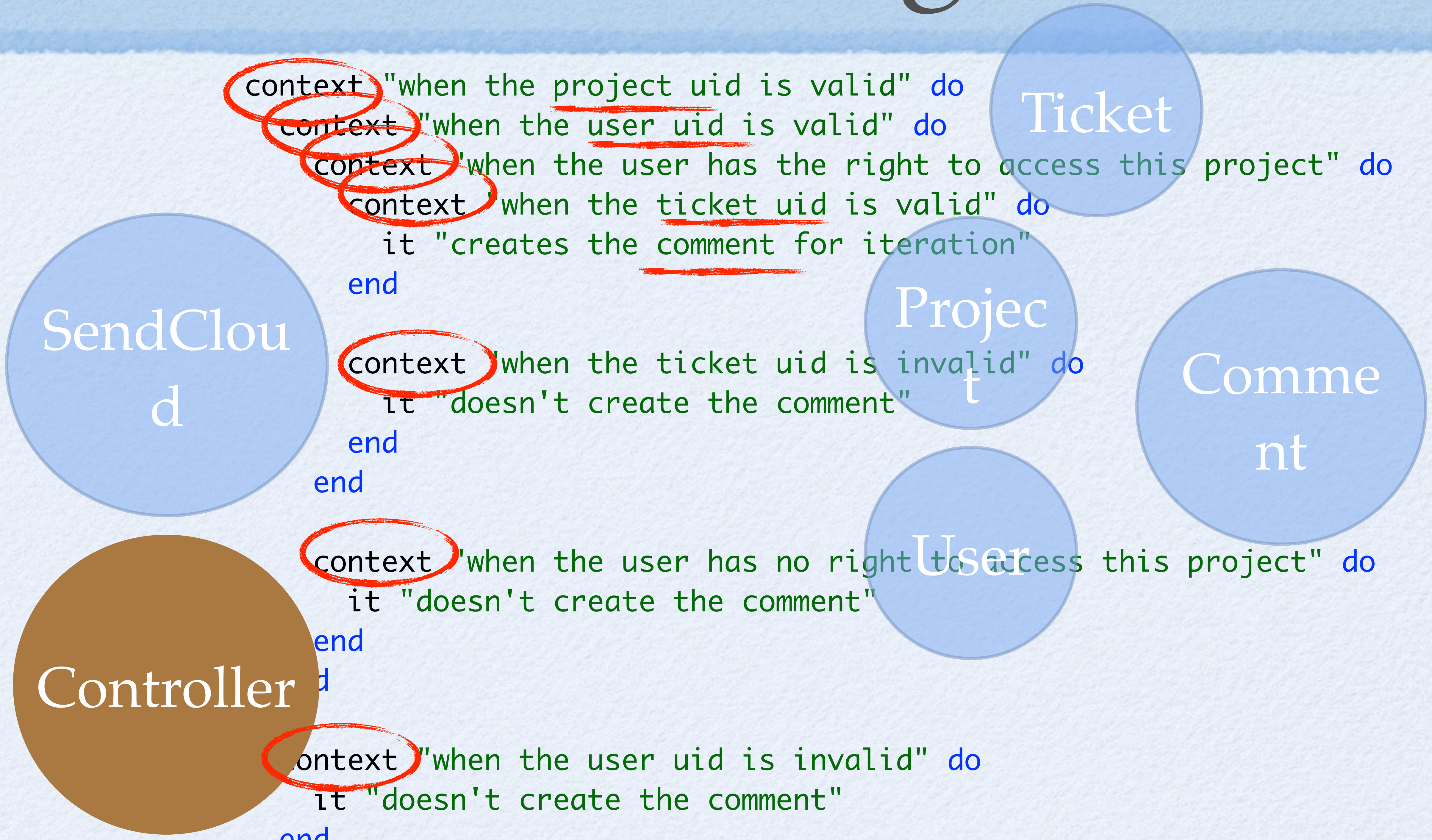
What do we get now?

```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

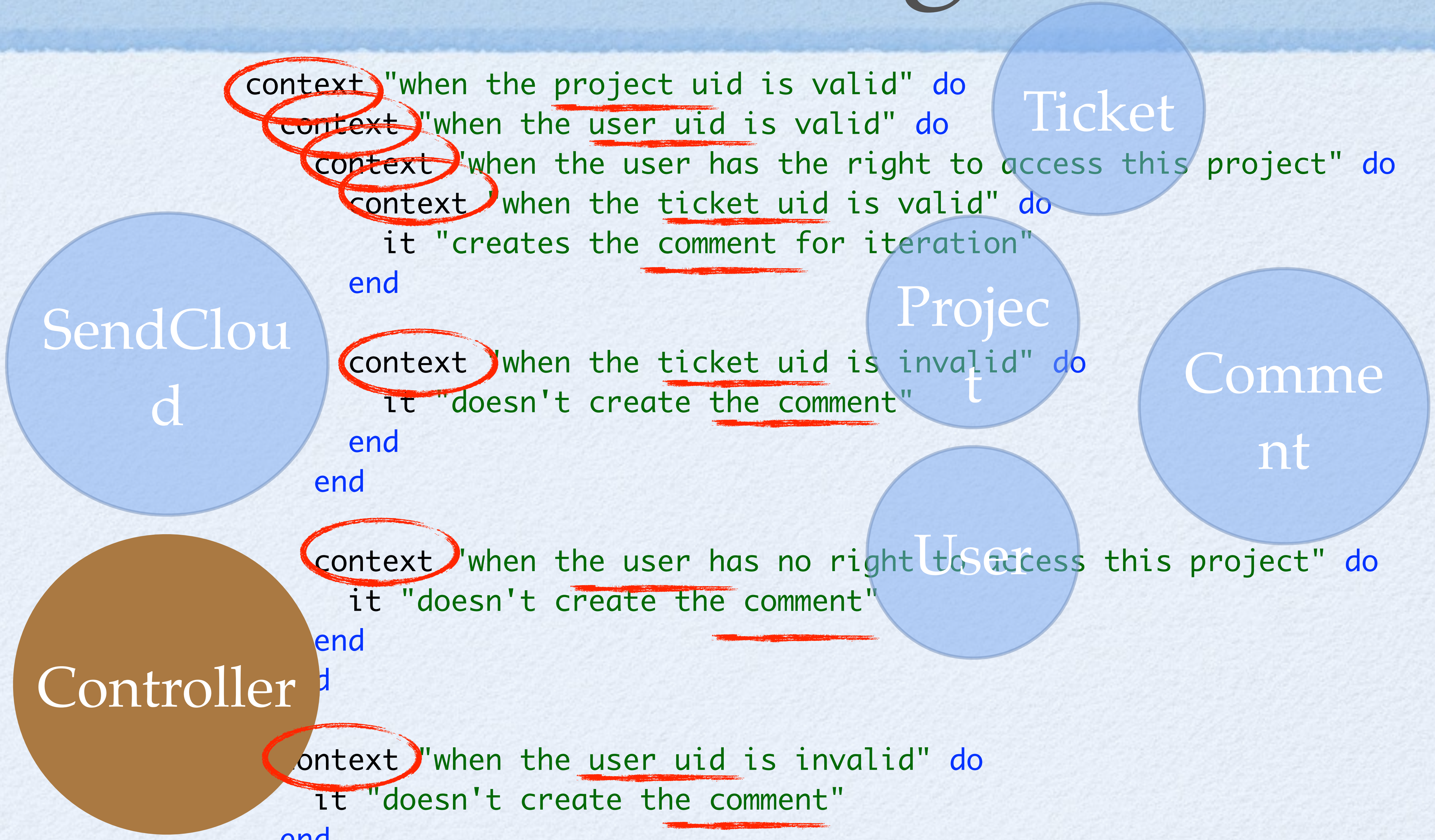


What do we get now?

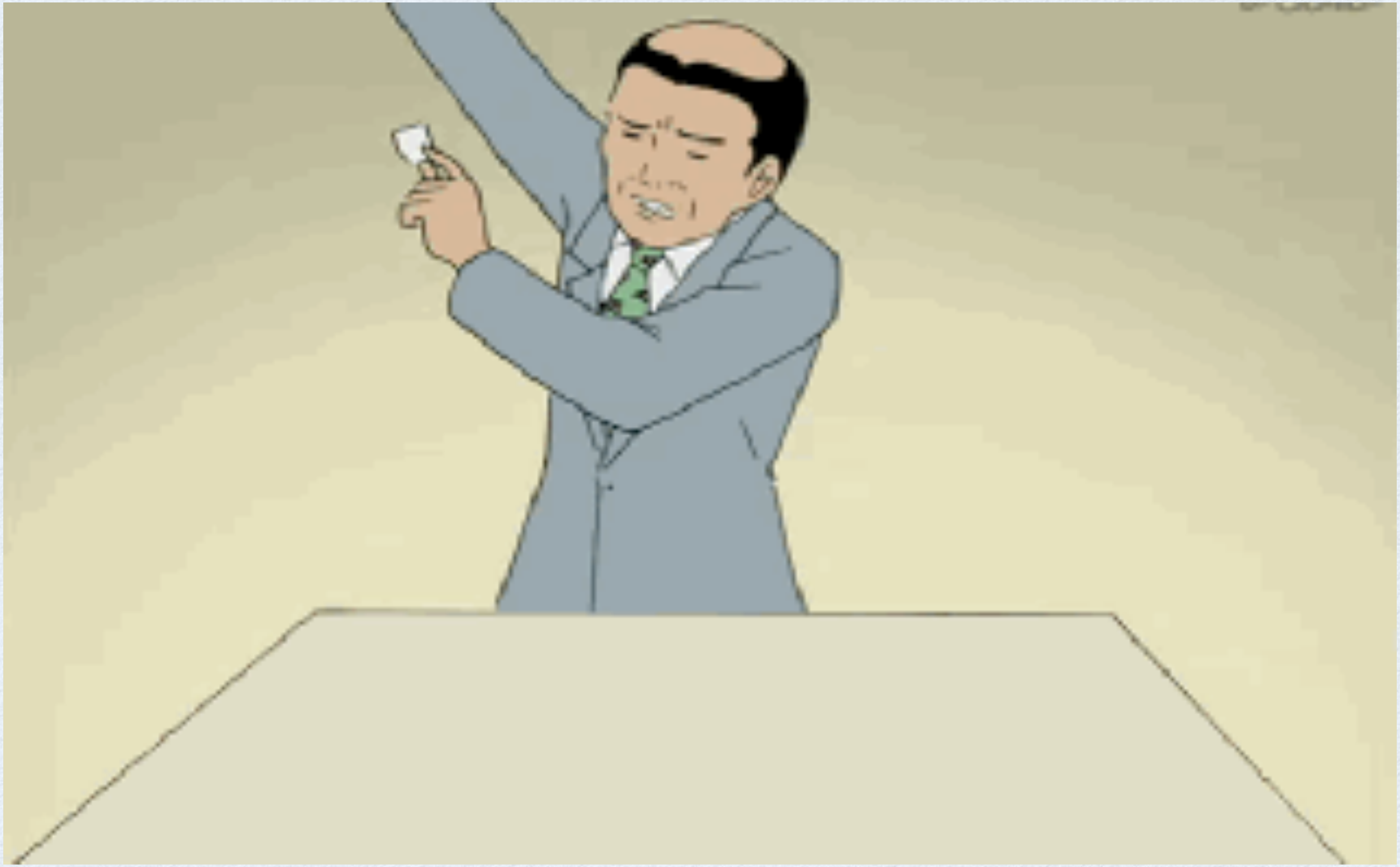
```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

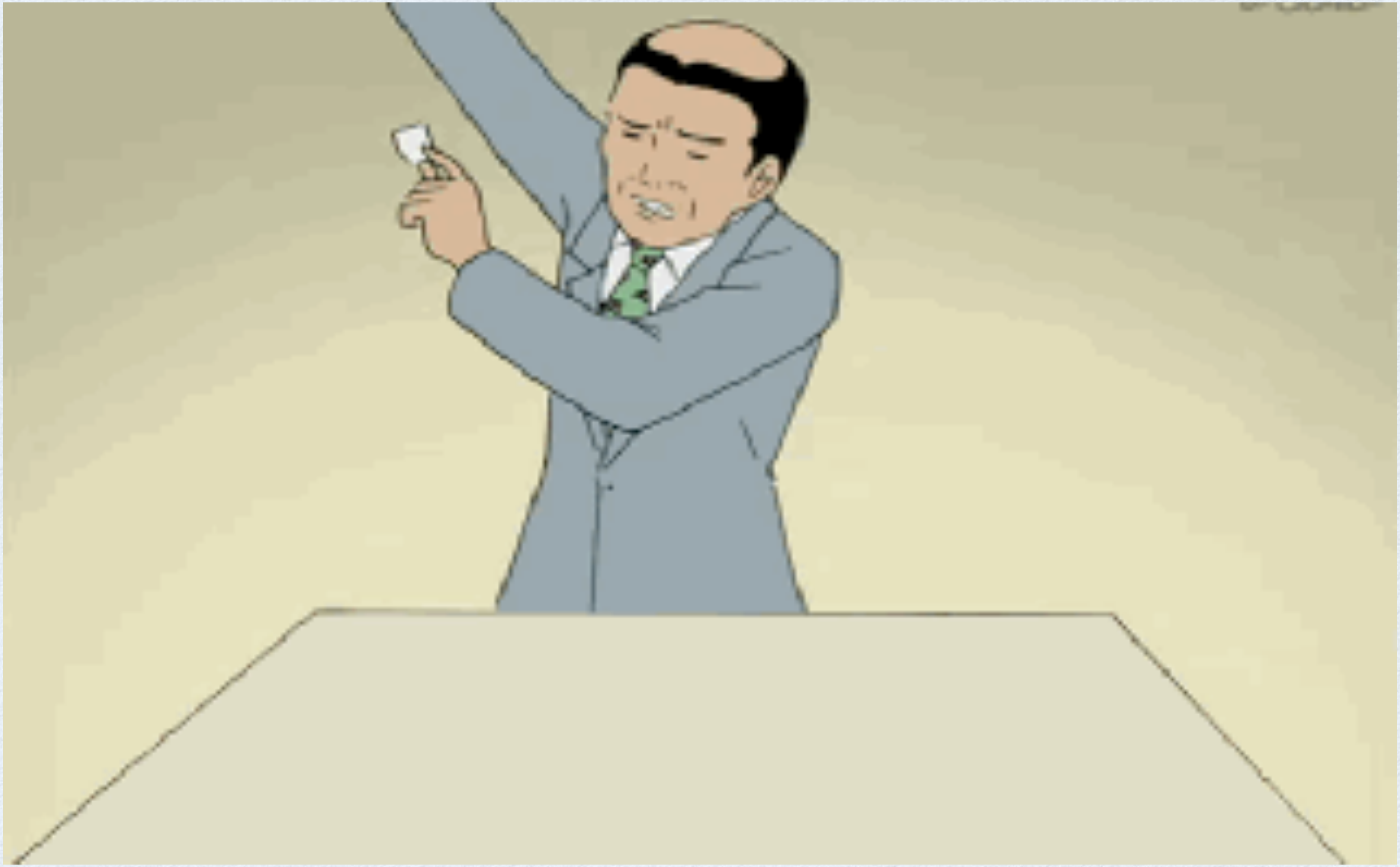


What do we get now?



```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```





No Test!

No Test!

```
def create
```

No Test!

```
def create
  if post_request_valid
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)

    if project.valid
      user = User.find(...)
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
          end
        else
          ...
        end
      else
        ...
      end
    else
      ....
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
          end
        else
          ...
        end
      else
        ...
      end
    else
      ....
```

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
          end
        else
          ...
        end
      else
        ...
      end
    else
      ....
```

Refactor it?

No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
          end
        else
          ...
        end
      else
        ...
      end
    else
      ....
```

~~Refactor it?~~ It is too late!

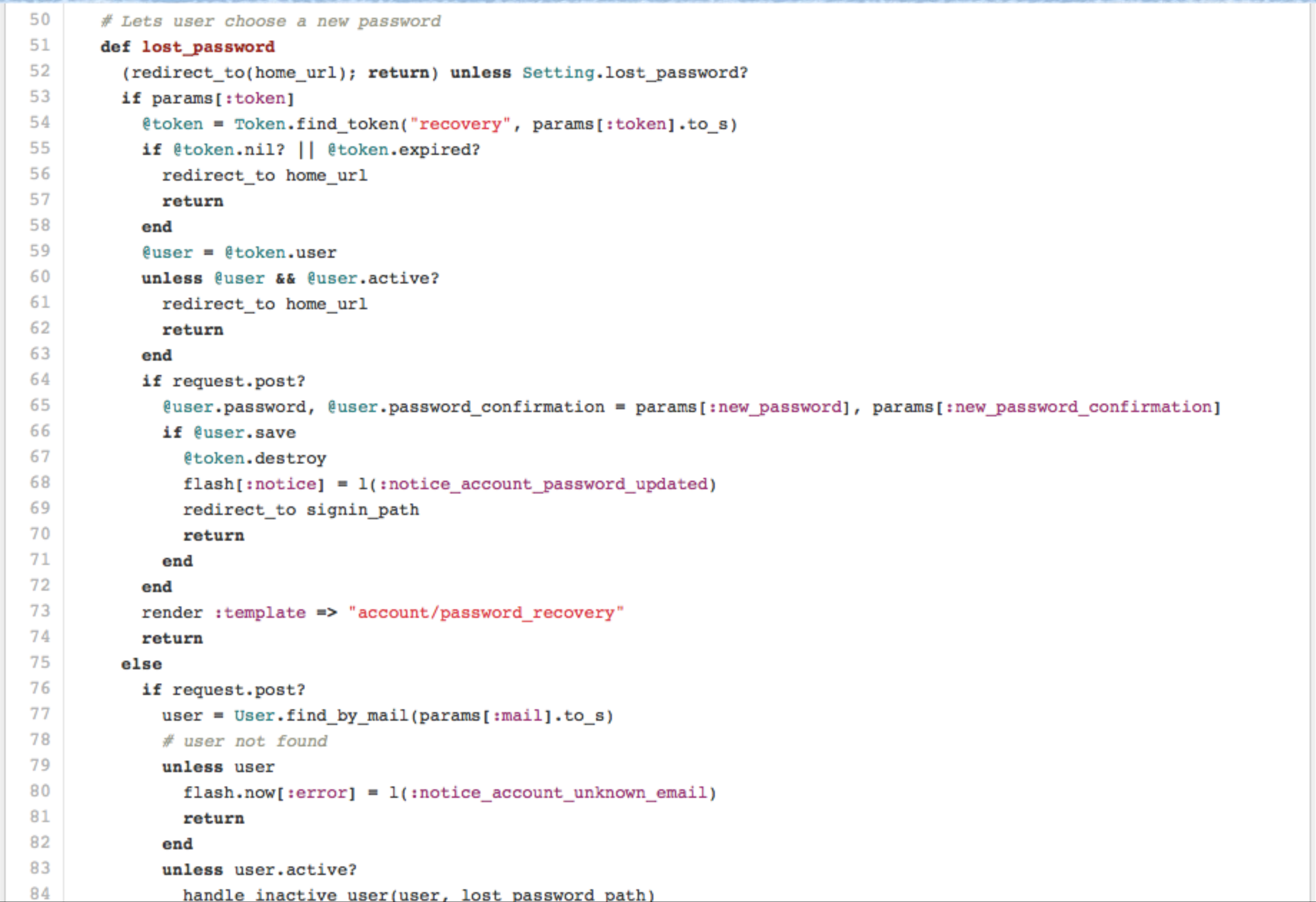
No Test!

```
def create
  if post_request_valid
    project = Project.find(...)
    if project.valid
      user = User.find(...)
      if user.valid
        if user.have_access_right
          ticket = Ticket.find(...)
          if ticket.valid
            Comment.create(...)
          end
        else
          ...
        end
      else
        ...
      end
    else
      ....
```

~~Refactor it?~~ It is too late!

Technical Debt

Redmine



```
# Lets user choose a new password
def lost_password
  (redirect_to(home_url); return) unless Setting.lost_password?
  if params[:token]
    @token = Token.find_token("recovery", params[:token].to_s)
    if @token.nil? || @token.expired?
      redirect_to home_url
      return
    end
    @user = @token.user
    unless @user && @user.active?
      redirect_to home_url
      return
    end
    if request.post?
      @user.password, @user.password_confirmation = params[:new_password], params[:new_password_confirmation]
      if @user.save
        @token.destroy
        flash[:notice] = l(:notice_account_password_updated)
        redirect_to signin_path
        return
      end
    end
    render :template => "account/password_recovery"
    return
  else
    if request.post?
      user = User.find_by_mail(params[:mail].to_s)
      # user not found
      unless user
        flash.now[:error] = l(:notice_account_unknown_email)
        return
      end
      unless user.active?
        handle_inactive_user(user, lost_password_path)
```

Let's go back here

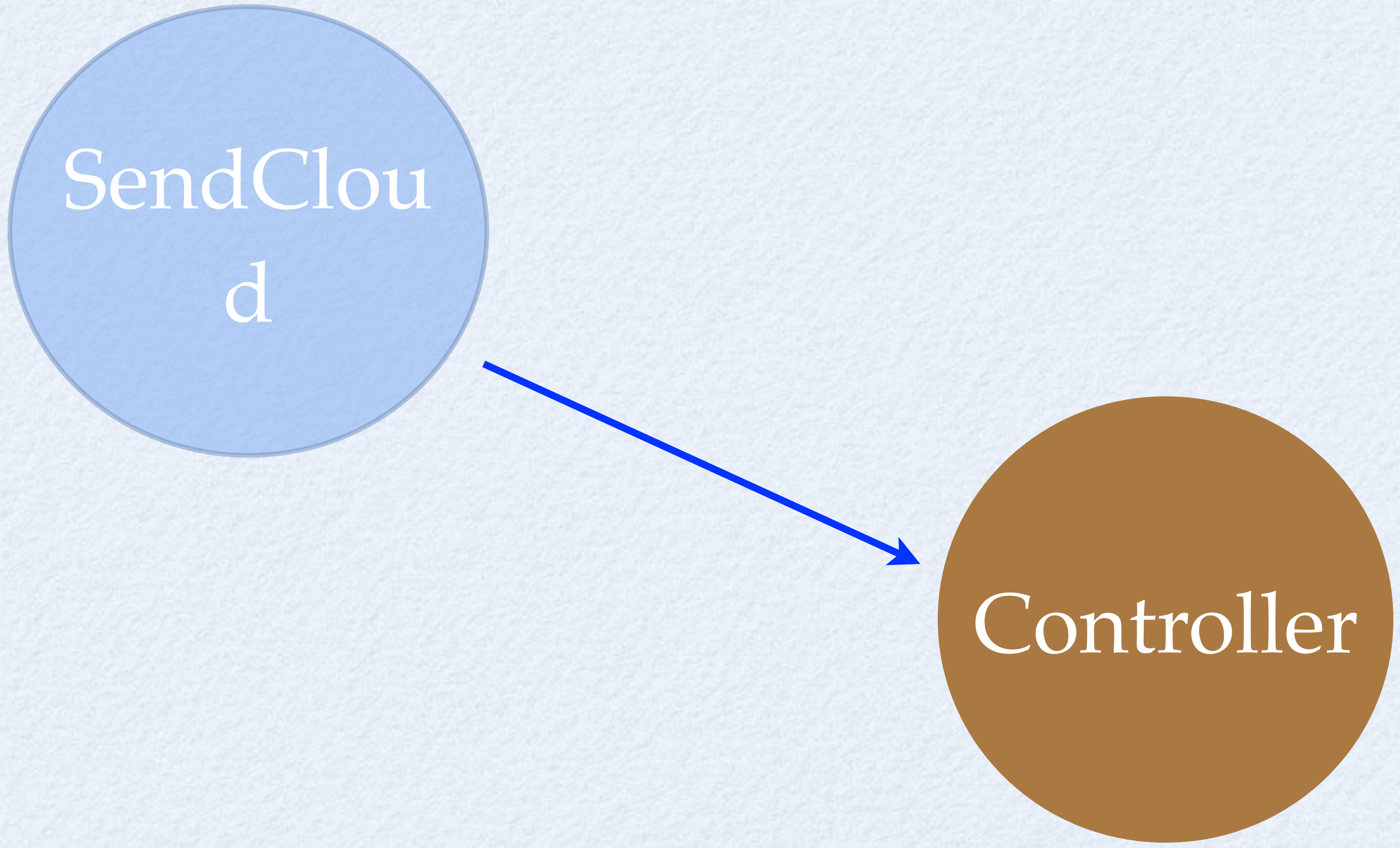
```
context "when the project uid is valid" do
  context "when the user uid is valid" do
    context "when the user has the right to access this project" do
      context "when the ticket uid is valid" do
        it "creates the comment for iteration"
      end

      context "when the ticket uid is invalid" do
        it "doesn't create the comment"
      end
    end

    context "when the user has no right to access this project" do
      it "doesn't create the comment"
    end
  end

  context "when the user uid is invalid" do
    it "doesn't create the comment"
  end
end

context "when the project uid is invalid" do
  it "doesn't create the comment"
end
```

Test reflect the design

```
describe "POST create" do
```

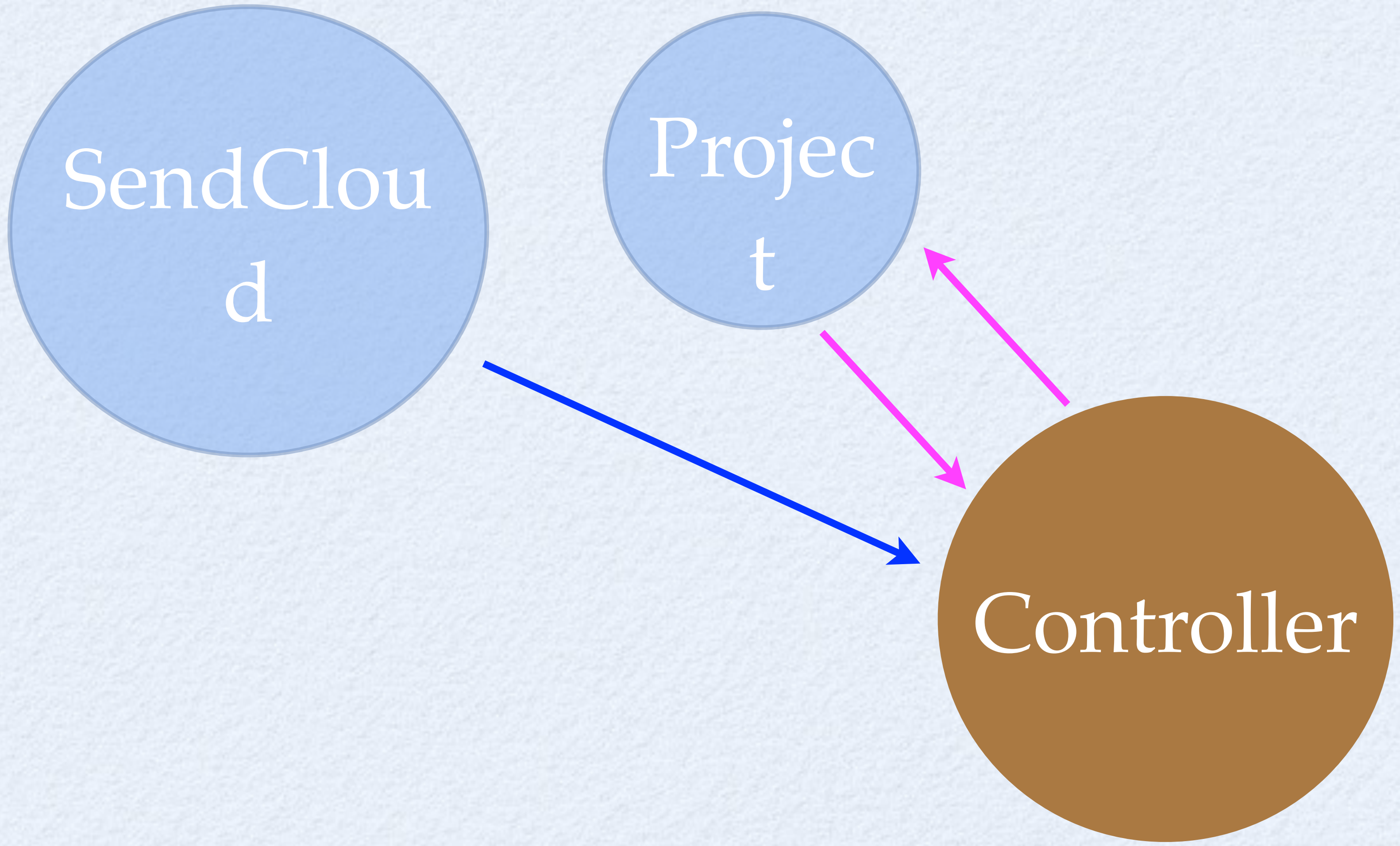
Test reflect the design

```
describe "POST create" do
  context "when the post request is a valid request" do
```



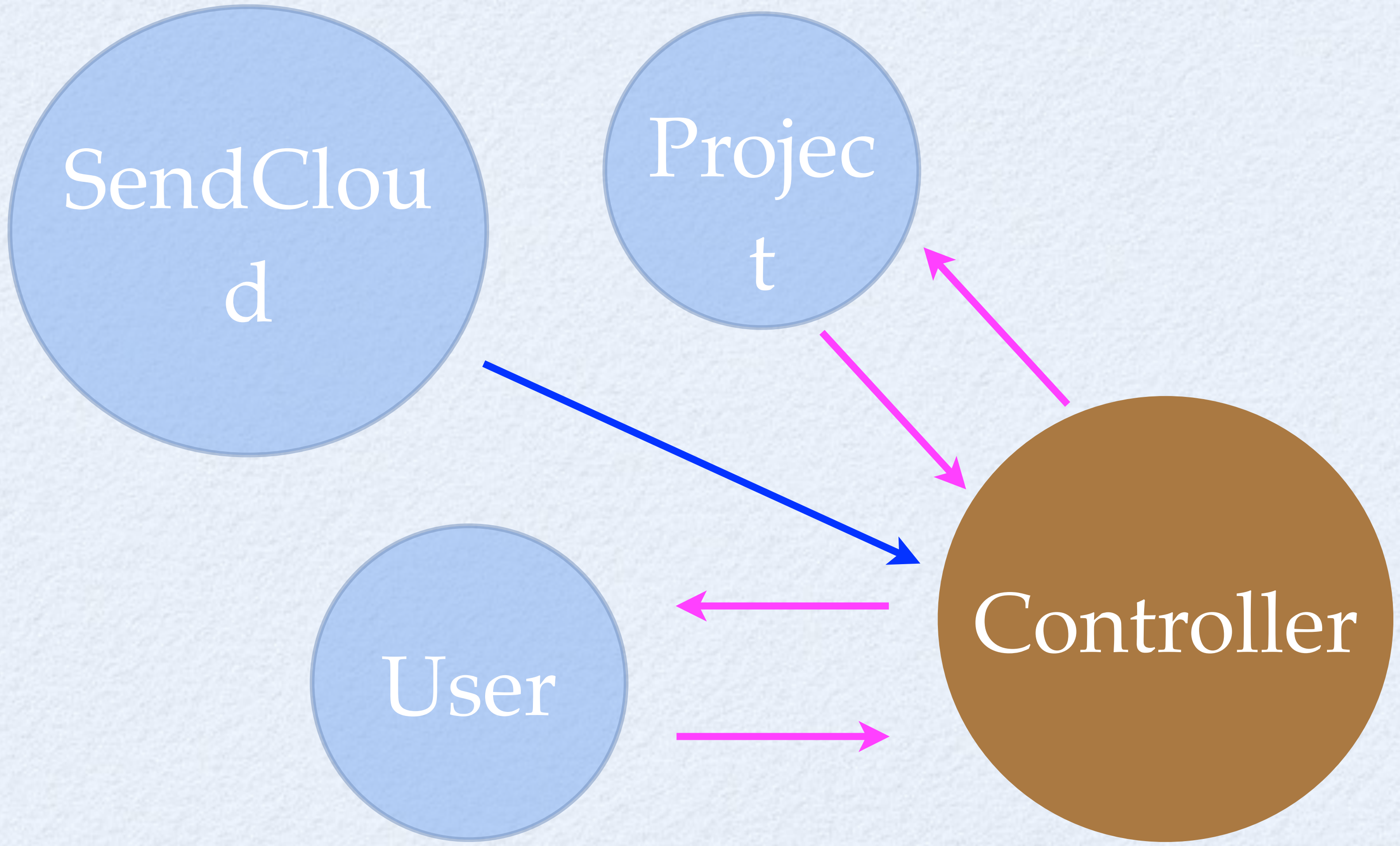
Test reflect the design

```
describe "POST create" do
 context "when the post request is a valid request" do
   context "when the project uid is valid" do
```



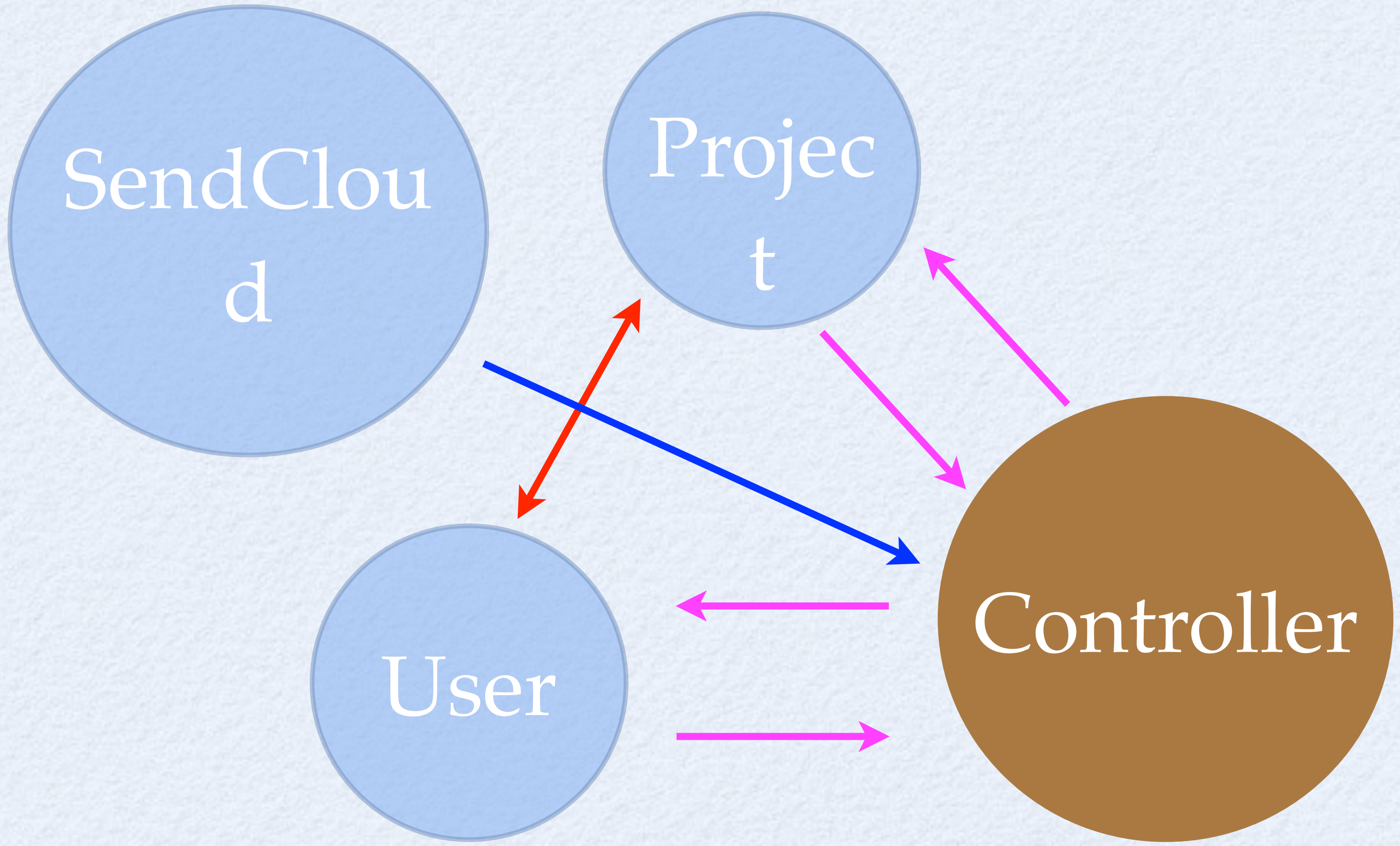
Test reflect the design

```
describe "POST create" do
 context "when the post request is a valid request" do
   context "when the project uid is valid" do
     context "when the user uid is valid" do
```



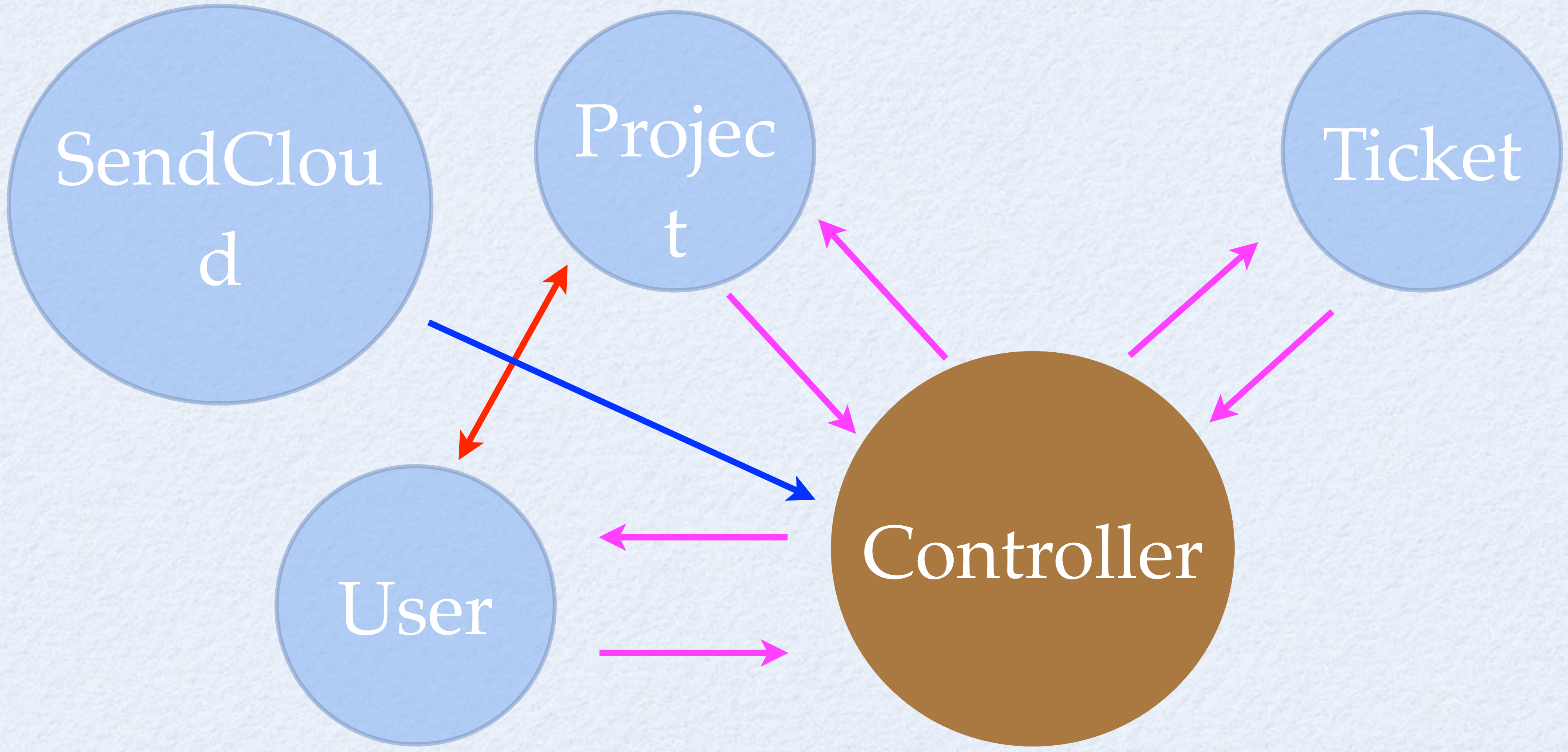
Test reflect the design

```
describe "POST create" do
 context "when the post request is a valid request" do
   context "when the project uid is valid" do
     context "when the user uid is valid" do
       context "when the user has the right to access this project" do
```



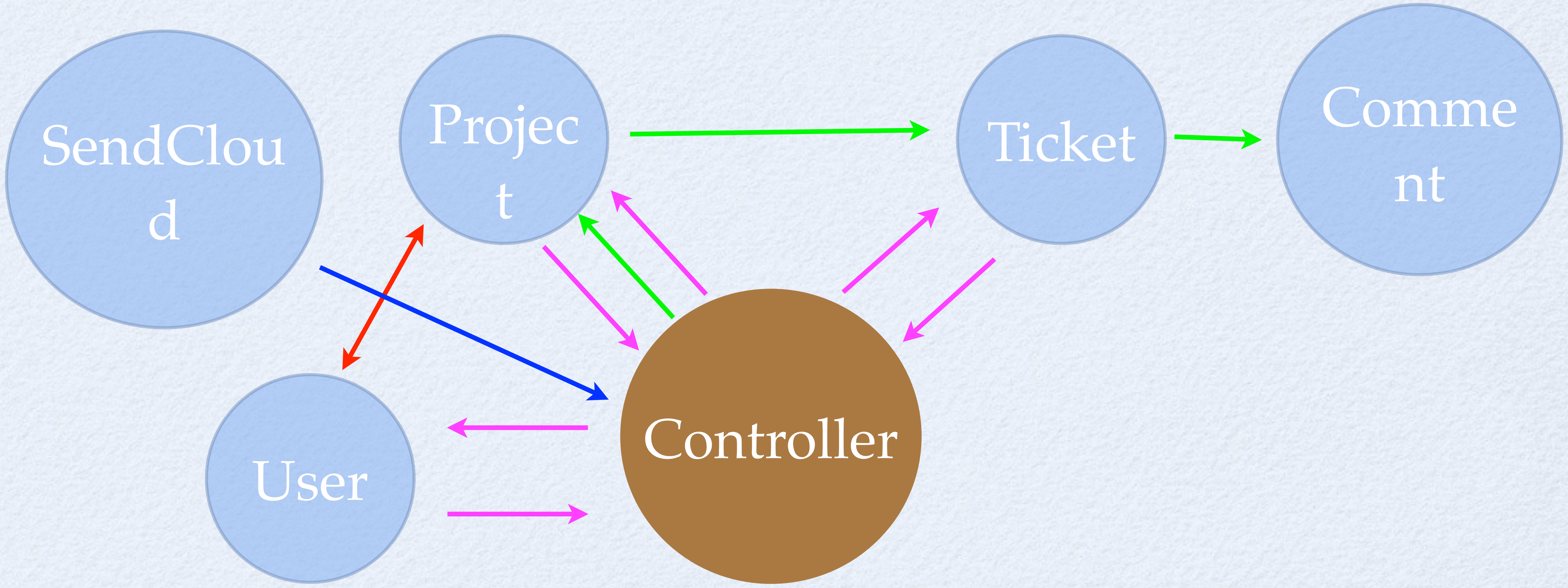
Test reflect the design

```
describe "POST create" do
 context "when the post request is a valid request" do
   context "when the project uid is valid" do
     context "when the user uid is valid" do
       context "when the user has the right to access this project" do
         context "when the ticket uid is valid" do
```



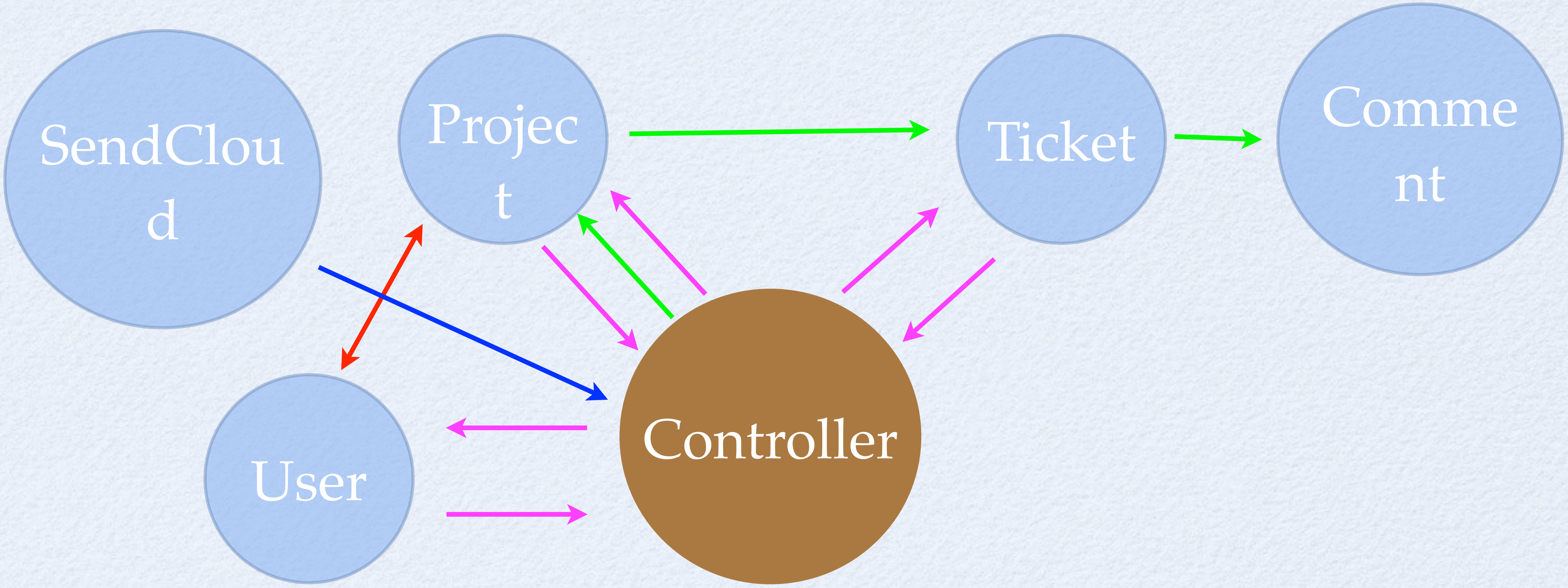
Test reflect the design

```
describe "POST create" do
 context "when the post request is a valid request" do
   context "when the project uid is valid" do
      context "when the user uid is valid" do
         context "when the user has the right to access this project" do
            context "when the ticket uid is valid" do
               it "creates the comment for iteration"
```

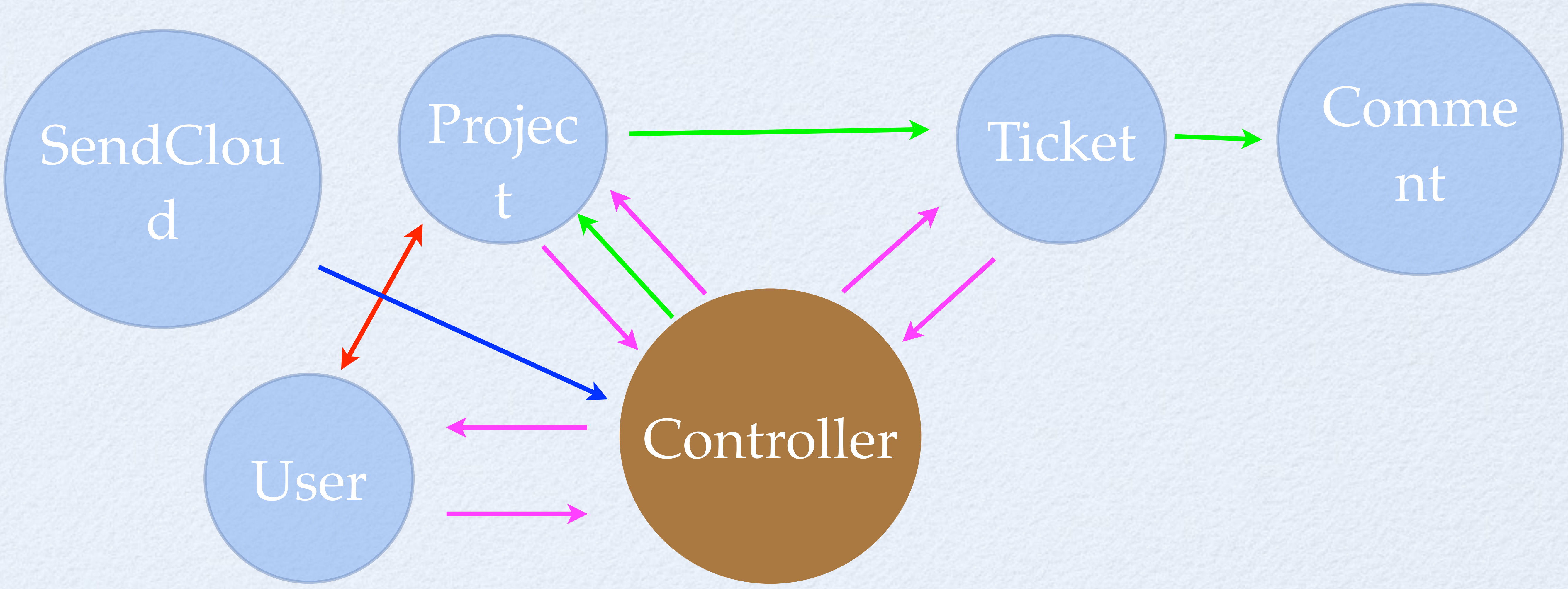


1. I didn't write any codes
2. What we found from test

Test reflect the design

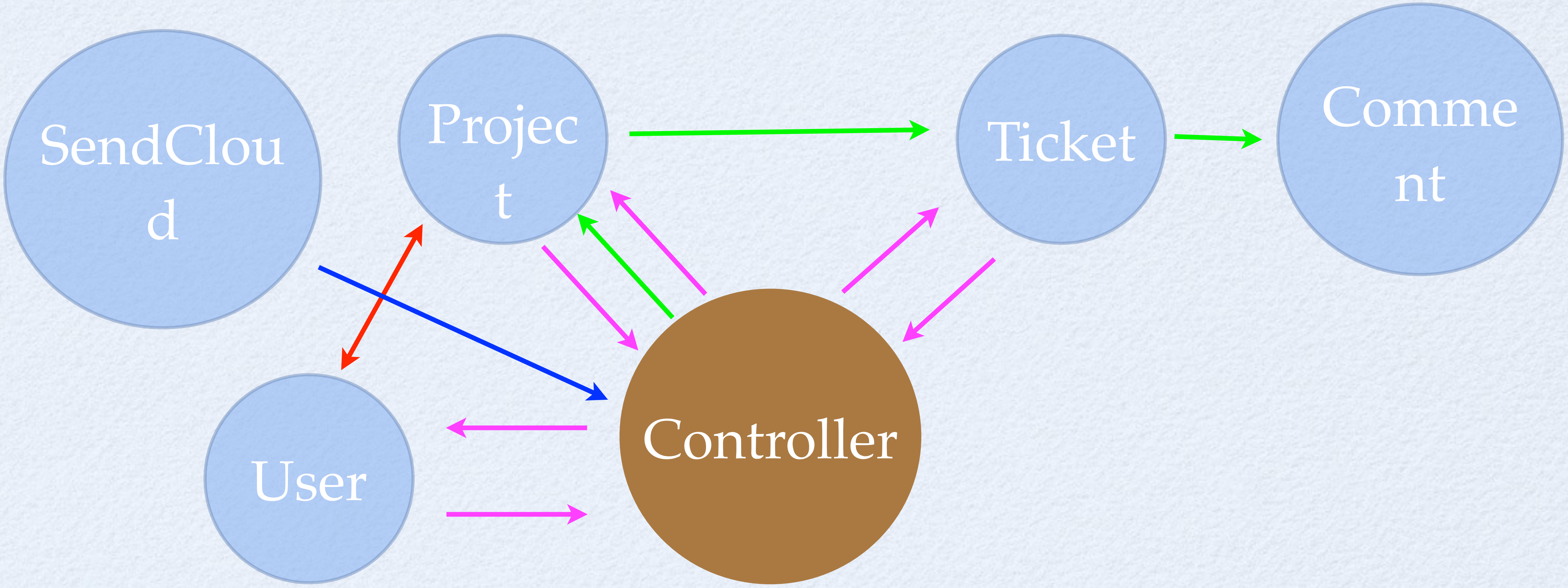


Test reflect the design



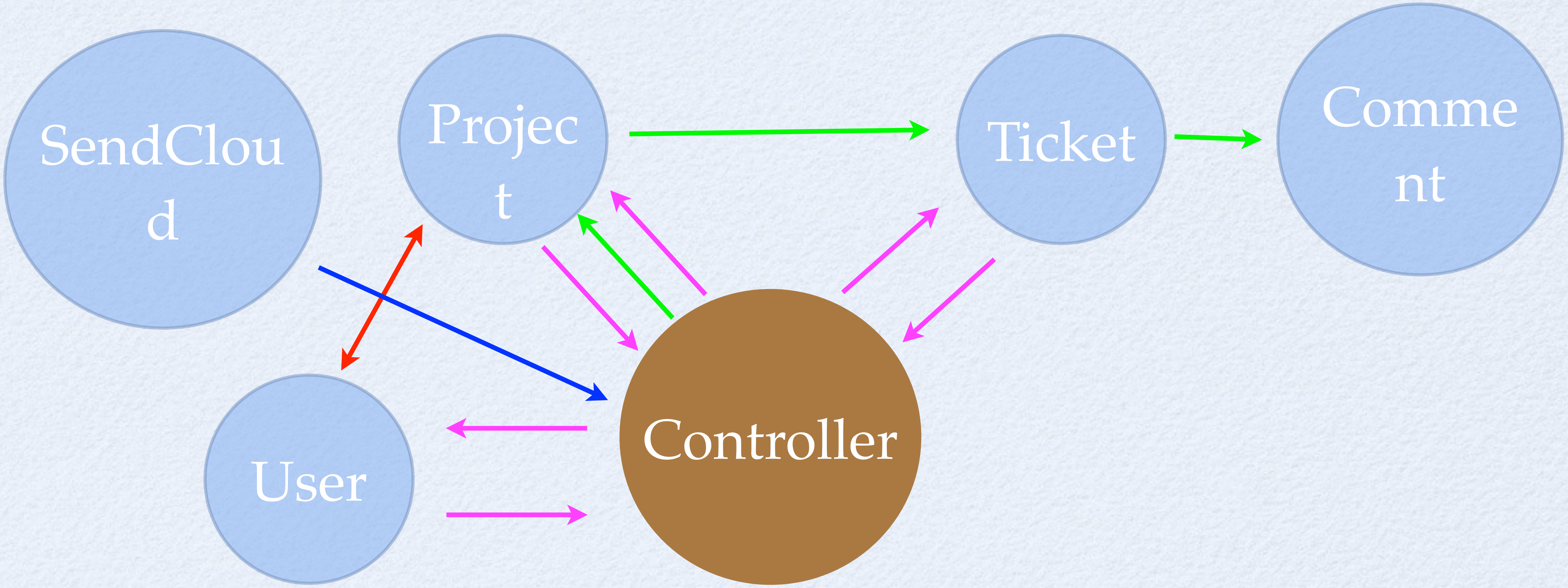
Structural Coupling!

Test reflect the design



Structural Coupling!
Doing too many things!

Test reflect the design



Structural Coupling!

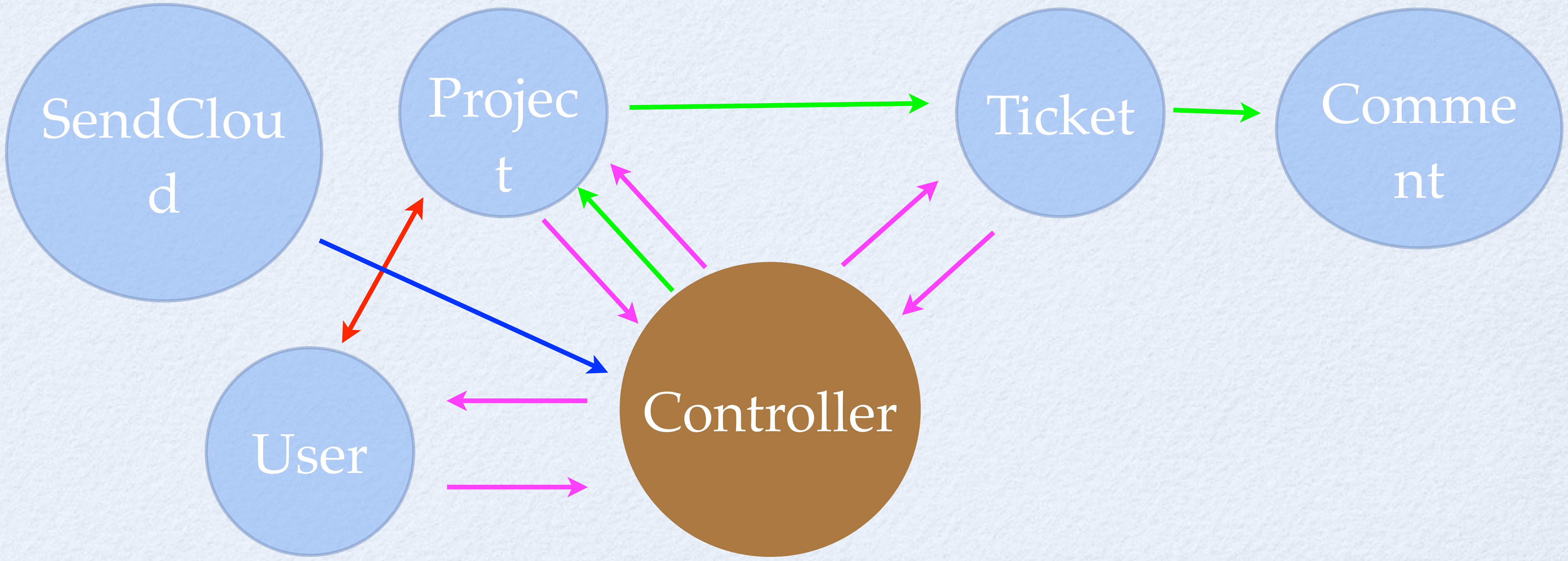
Doing too many things!

Knowing too much details!

Let’s re-design it from test!

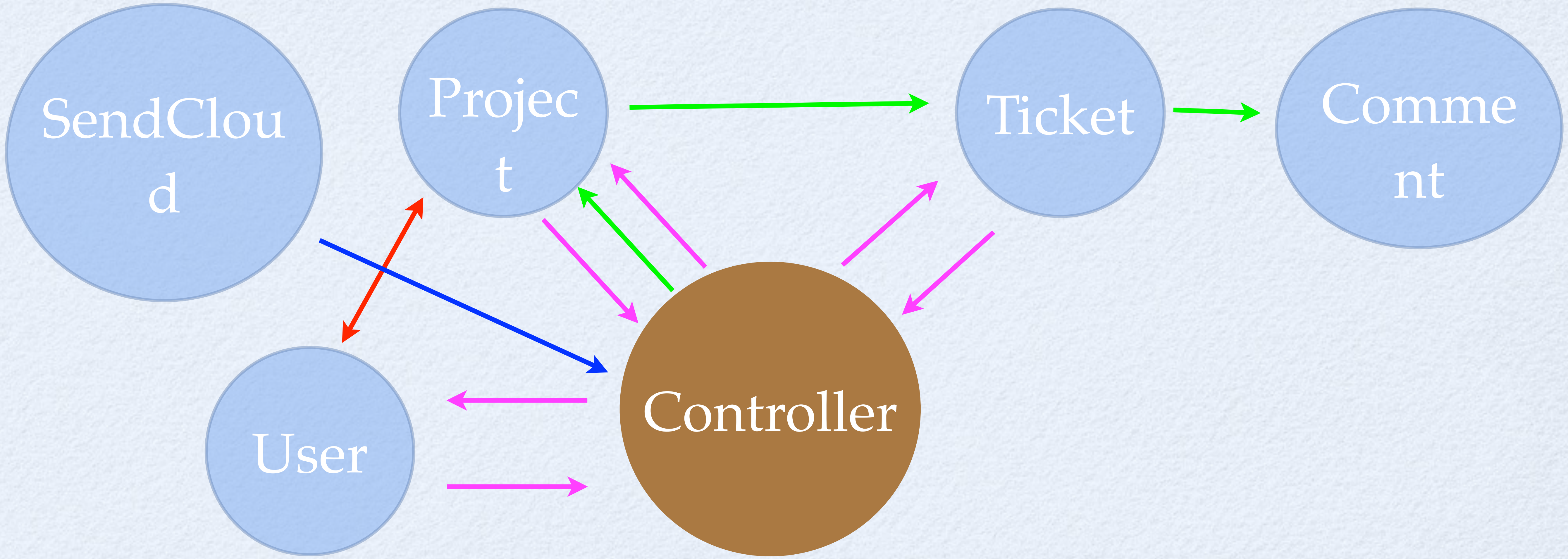
Re-Design From Test

```
context "when the post request is a valid request" do
  context "when the project uid is valid" do
    context "when the user uid is valid" do
      context "when the user has the right to access this project" do
        context "when the ticket uid is valid" do
          it "creates the comment for iteration"
```



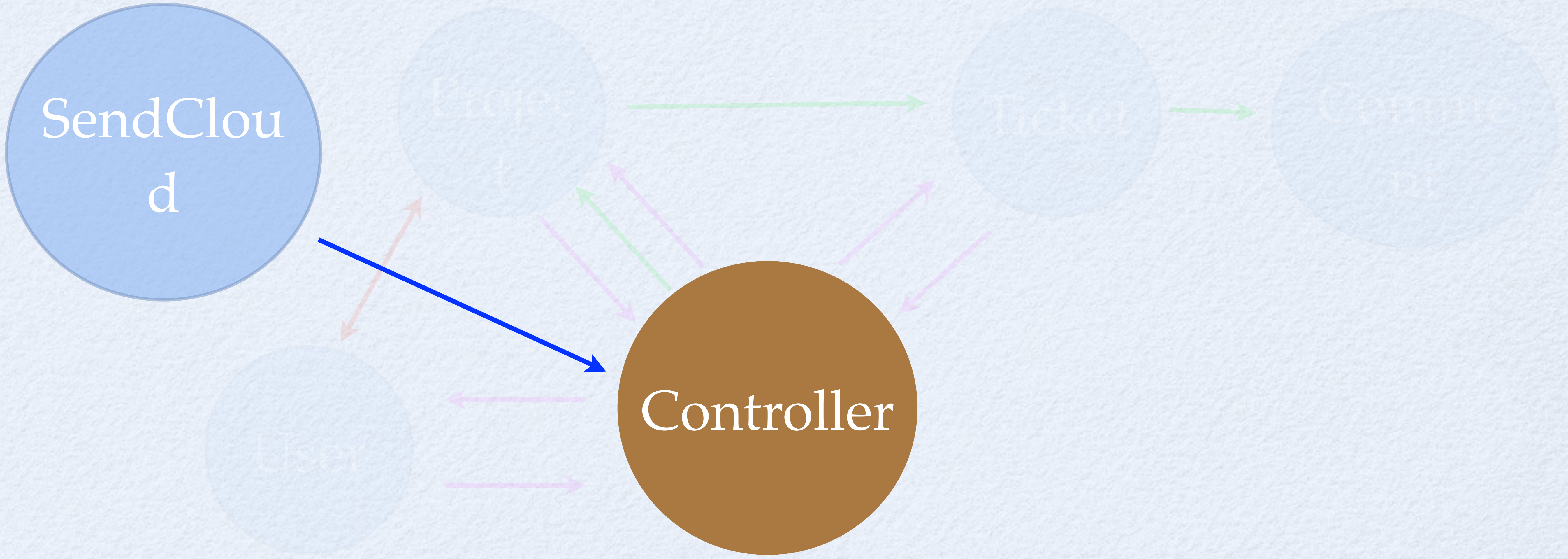
Re-Design From Test

```
context "when the post request is a valid request" do
  context "when the project uid is valid" do
    context "when the user uid is valid" do
      context "when the user has the right to access this project" do
        context "when the ticket uid is valid" do
          it "creates the comment for iteration"
```



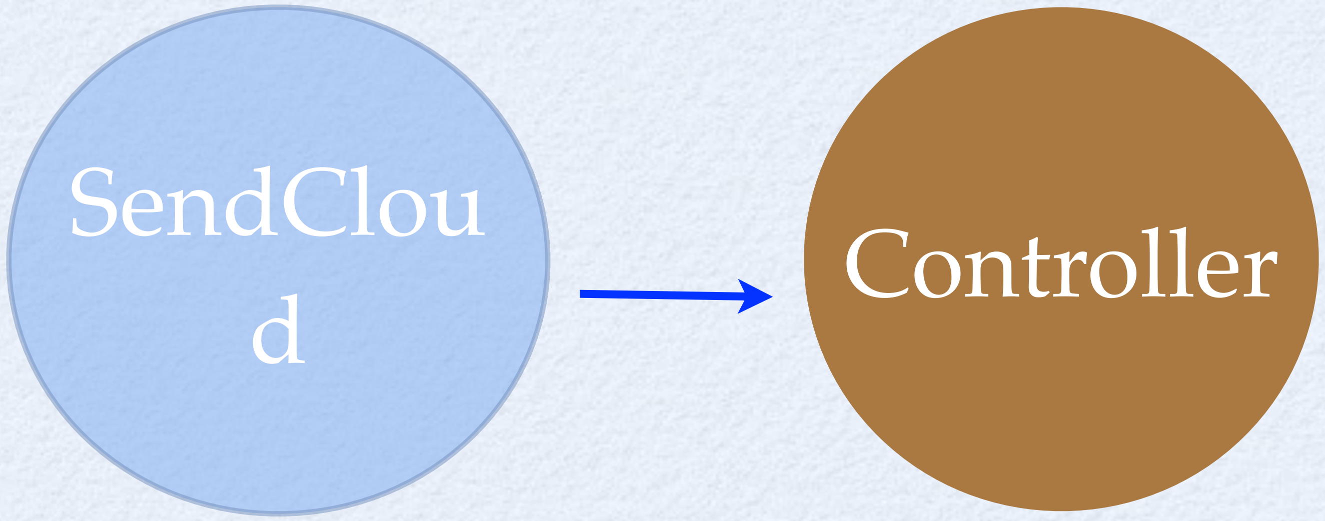
Re-Design From Test

```
context "when the post request is a valid request" do
  context "when the project uid is valid" do
    context "when the user uid is valid" do
      context "when the user has the right to access this project" do
        context "when the ticket uid is valid" do
          it "creates the comment for iteration"
```



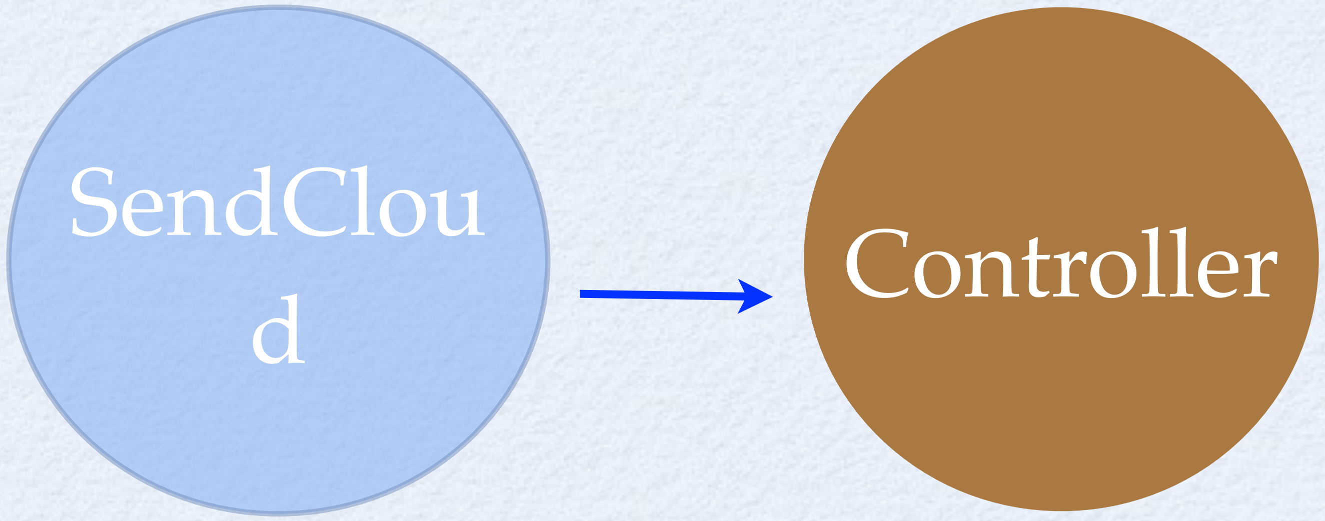
Re-Design

```
context "when the post request is a valid request" do
```



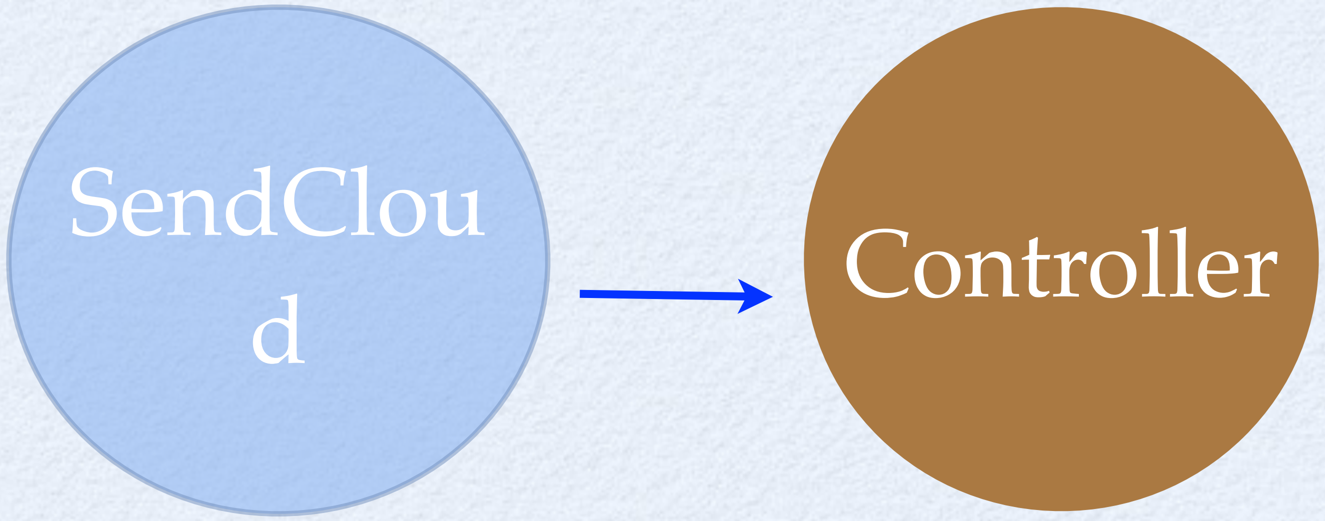
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do





  end
```



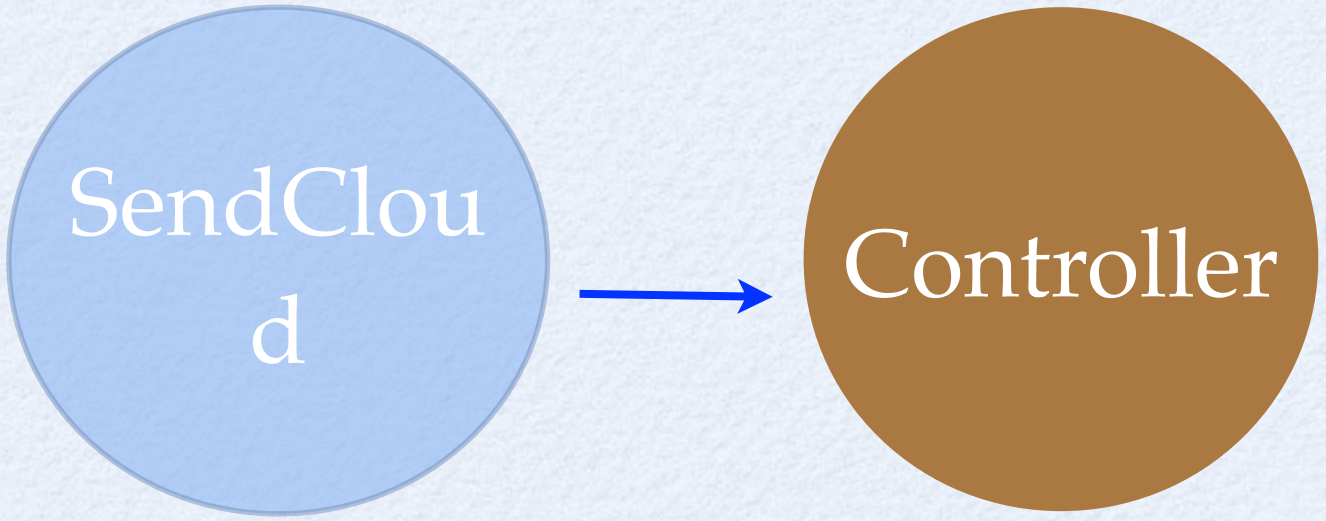
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do




  end
```



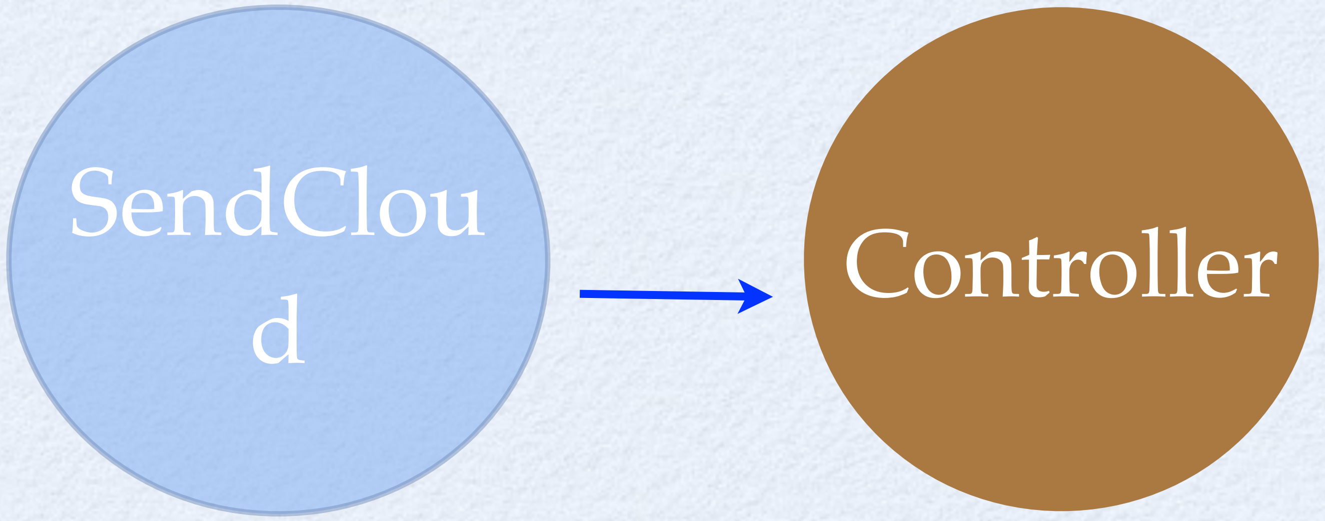
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do

     email_handler = EmailHandler.any_instance




  end
```



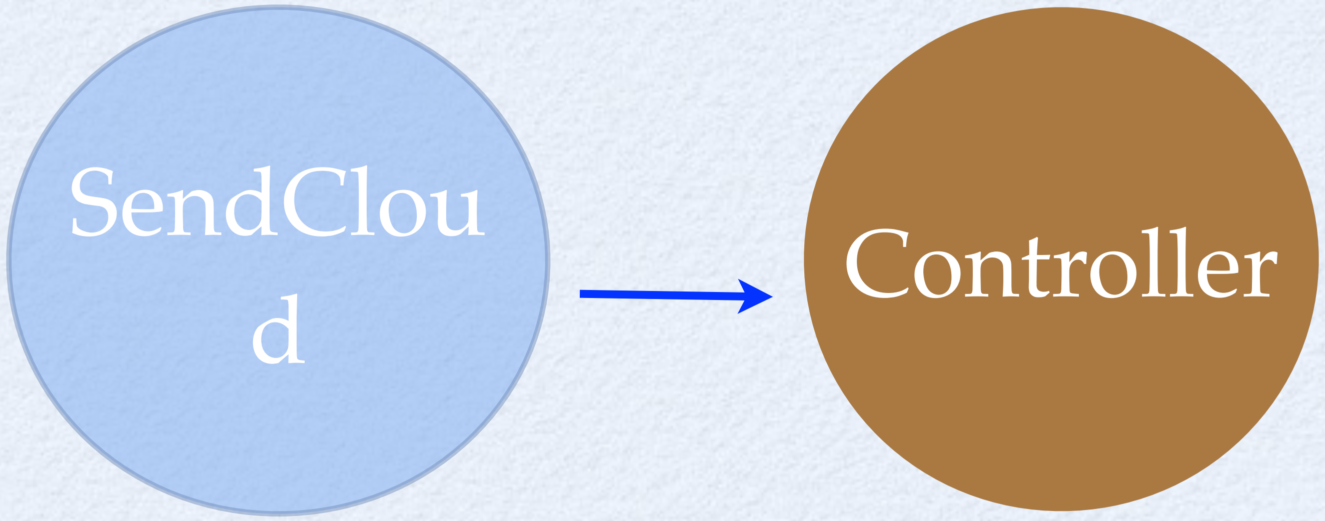
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do
    email_handler = EmailHandler.any_instance



  end
```



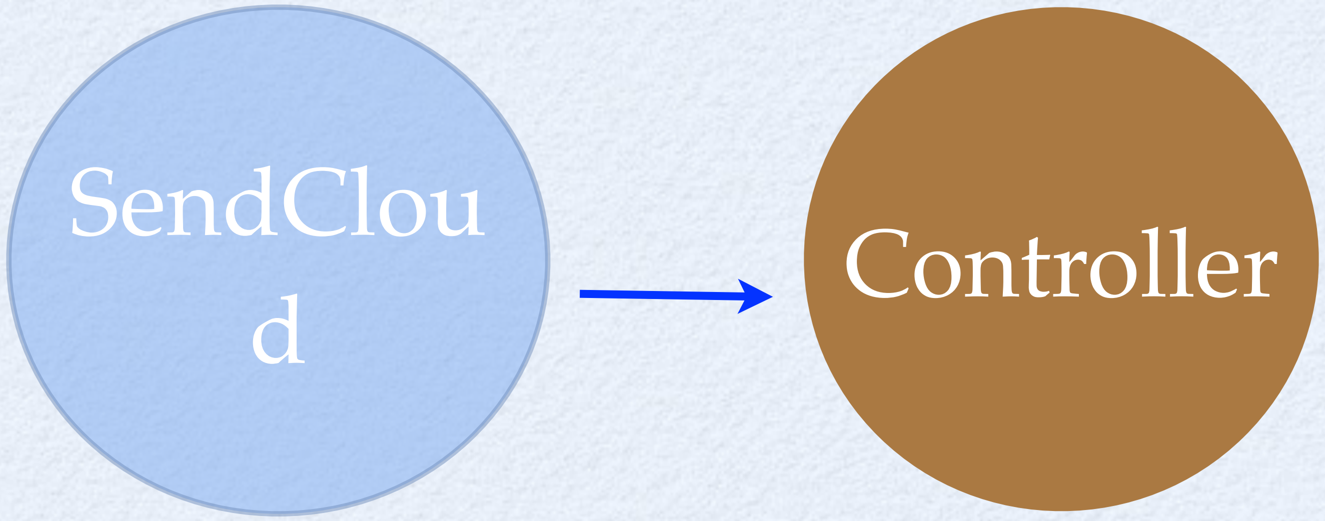
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do

    email_handler = EmailHandler.any_instance

    email_handler.should_receive(:handle)



  end
```



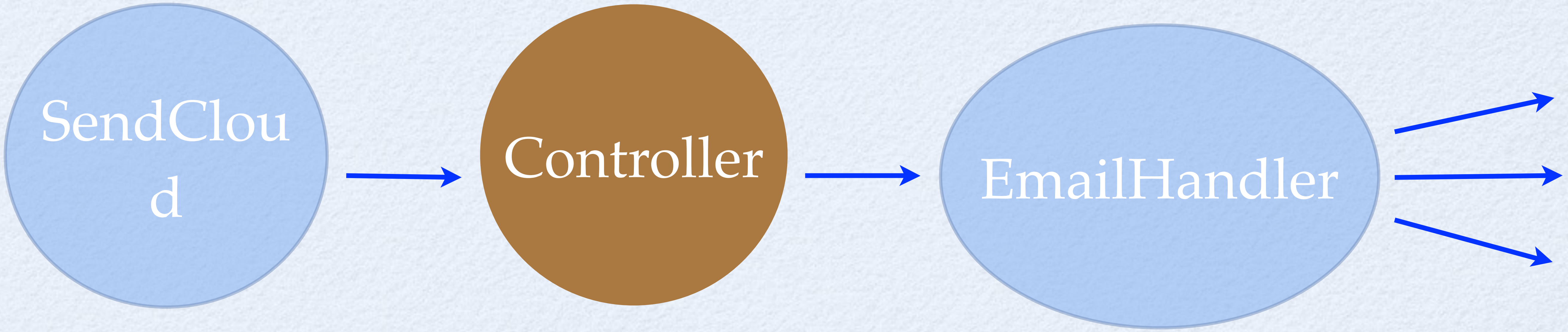
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do

    email_handler = EmailHandler.any_instance

    email_handler.should_receive(:handle)

    post :create, timestamp: timestamps, token: token, signature:
    signature
  end
```



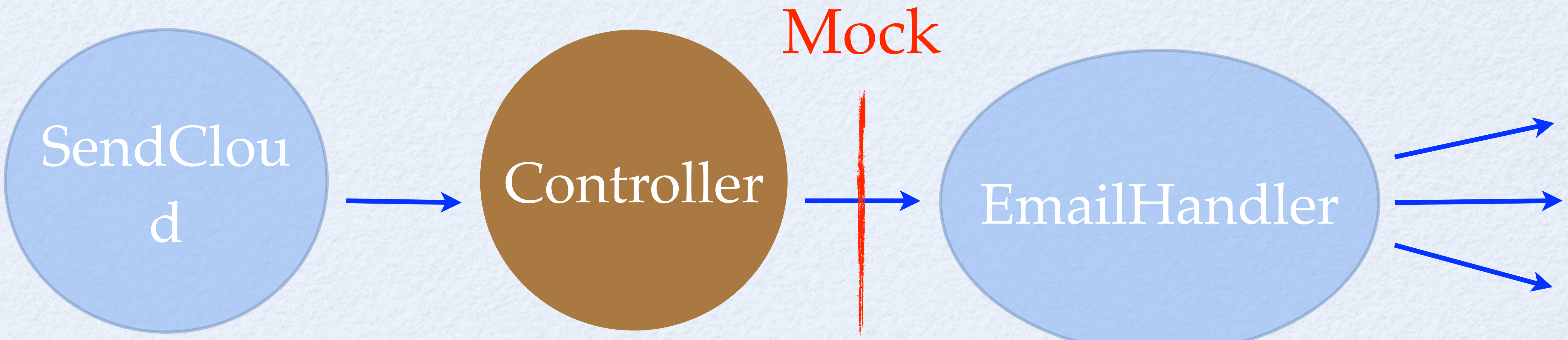
Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do
    email_handler = EmailHandler.any_instance
    email_handler.should_receive(:handle)
    post :create, timestamp: timestamps, token: token, signature:
    signature
  end
```



Re-Design

```
context "when the post request is a valid request" do
  it "tells email handler to handle the request" do

    email_handler = EmailHandler.any_instance

    email_handler.should_receive(:handle)

    post :create, timestamp: timestamps, token: token, signature:
    signature
  end
```



Re-Design Controller

```
class EmailRepliesController < ApplicationController

  def create

    if post_request_authenticated
      EmailHandler.new(params).handle
      head(200)
    else
      head(422)
    end
  end

  def post_request_authenticated
    ...
  end
end
```

Re-Design Controller

```
class EmailRepliesController < ApplicationController

  def create

    if post_request_authenticated
      EmailHandler.new(params).handle
      head(200)
    else
      head(422)
    end
  end

  def post_request_authenticated
    ...
  end
end
```

Test

```
email_handler = EmailHandler.any_instance

email_handler.should_receive(:handle)
```

Re-Design Controller

```
class EmailRepliesController < ApplicationController

  def create

    if post_request_authenticated
      EmailHandler.new(params).handle
      head(200)
    else
      head(422)
    end
  end

  def post_request_authenticated
    ...
  end
end
```

Test

```
email_handler = EmailHandler.any_instance

email_handler.should_receive(:handle)
```

New Class EmailHandler

```
EmailHandler.new(params).handle
```

New Class EmailHandler

```
EmailHandler.new(params).handle

class EmailHandler
  def initialize(params)
  end

  def handle
  end
end
```

New Class EmailHandler

```
EmailHandler.new(params).handle
```

```
class EmailHandler
  def initialize(params)
  end

  def handle
  end
end
```

```
describe EmailHandler do
  describe "#initialize" do
  end

  describe "#handle" do
  end
end
```

New Class EmailHandler

```
EmailHandler.new(params).handle
```

```
class EmailHandler
  def initialize(params)
  end

  def handle
  end
end
```

```
describe EmailHandler do
  describe "#initialize" do
  end

  describe "#handle" do
  end
end
```

```
describe "#handle" do







  end
```

New Class EmailHandler

```
EmailHandler.new(params).handle
```

```
class EmailHandler
  def initialize(params)
  end

  def handle
  end
end
```

```
describe EmailHandler do
  describe "#initialize" do
  end

  describe "#handle" do
  end
end
```

```
describe "#handle" do
  context "when the project uid is valid" do
    context "when the user email is valid" do
      context "when the user has the right to access this project"
do
        context "when the ticket uid is valid" do
          ....

  end
```

EmailHandler#handle

```
  describe "#handle" do
    context "when the project uid is valid" do
      context "when the user email is valid" do
        context "when the user has the right to access this project" do
          context "when the ticket uid is valid" do
            ....

     end
```

EmailHandler#handle

```
  describe "#handle" do
    context "when the project uid is valid" do
      context "when the user email is valid" do
        context "when the user has the right to access this project"
do
          context "when the ticket uid is valid" do
            ....

    end
```

EmailHandler#handle

```
  describe "#handle" do
    context "when the project uid is valid" do
      context "when the user email is valid" do
        context "when the user has the right to access this project"
do
          context "when the ticket uid is valid" do
            ....

    end
```

```
        describe "#handle" do
          context "when it is a valid email" do
            it "creates a comment"
          end

          context "when it is a invalid email" do
            it "doesn't create a comment"
          end
        end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do


    it "creates a comment" do


    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }

    it "creates a comment" do


    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }

    it "creates a comment" do


    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }
    before { email_handler.stub(:valid_email?).and_return(true) }
    it "creates a comment" do


    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }
    before { email_handler.stub(:valid_email?).and_return(true) }
    it "creates a comment" do
      email_hander.handle

    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }
    before { email_handler.stub(:valid_email?).and_return(true) }
    it "creates a comment" do
      email_hander.handle
      Comment.count.should == 1
    end
  end
```

EmailHandler#handle

```
describe "#handle" do
  context "when it is a valid email" do
    let(:email_handler) { EmailHandler.new(....) }
    before { email_handler.stub(:valid_email?).and_return(true) }
    it "creates a comment" do
      email_hander.handle
      Comment.count.should == 1
    end
  end


  context "when it is a invalid email" do
    let(:email_handler) { EmailHandler.new(....) }
    before { email_handler.stub(:valid_email?).and_return(false) }
    it "doesn't create a comment" do
      email_hander.handle
      Comment.count.should == 0
    end
  end
```

EmailHandler#handle

```ruby
class EmailHandler
  def initialize
  end

  def handle
    if valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email information.
  end
end
```

EmailHandler#handle

```ruby
class EmailHandler
  def initialize
  end

  def handle
    if valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email information.
  end
end
```

EmailHandler#handle

```
class EmailHandler
  def initialize
  end

  def handle
    if valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email information.
  end
end
```

EmailHandler#handle

```ruby
class EmailHandler
  def initialize
  end

  def handle
    if valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email information.
  end
end
```

```ruby
describe "#valid_email?" do

end
```

EmailHandler#handle

```
class EmailHandler
  def initialize
  end

  def handle
    if valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email information.
  end
end
```

```
describe "#valid_email?" do

  context "all objects is valid
    #true

  context "invalid project"
    #false

  context "invalid user"
    #false

  context "invalid ticket"
    #false
end
```

What we got

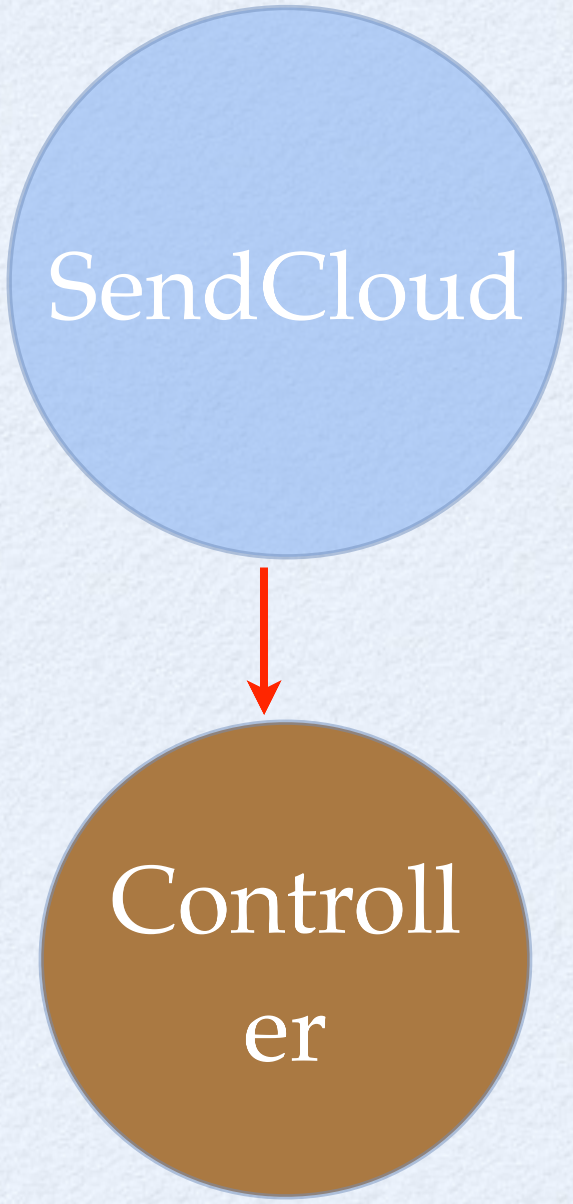
```
class EmailRepliesController < AC

  def create
    if post_request_authenticated?
      EmailHandler.new(params).handle
      head(200)
    else
      head(422)
    end
  end

  def post_request_authenticated?
    ...
  end
end
```

```
class EmailHandler
  def initialize
  end

  def handle
    if self.valid_email?
      #create comment
    end
  end

  def valid_email?
    #validate the email.
  end
end
```

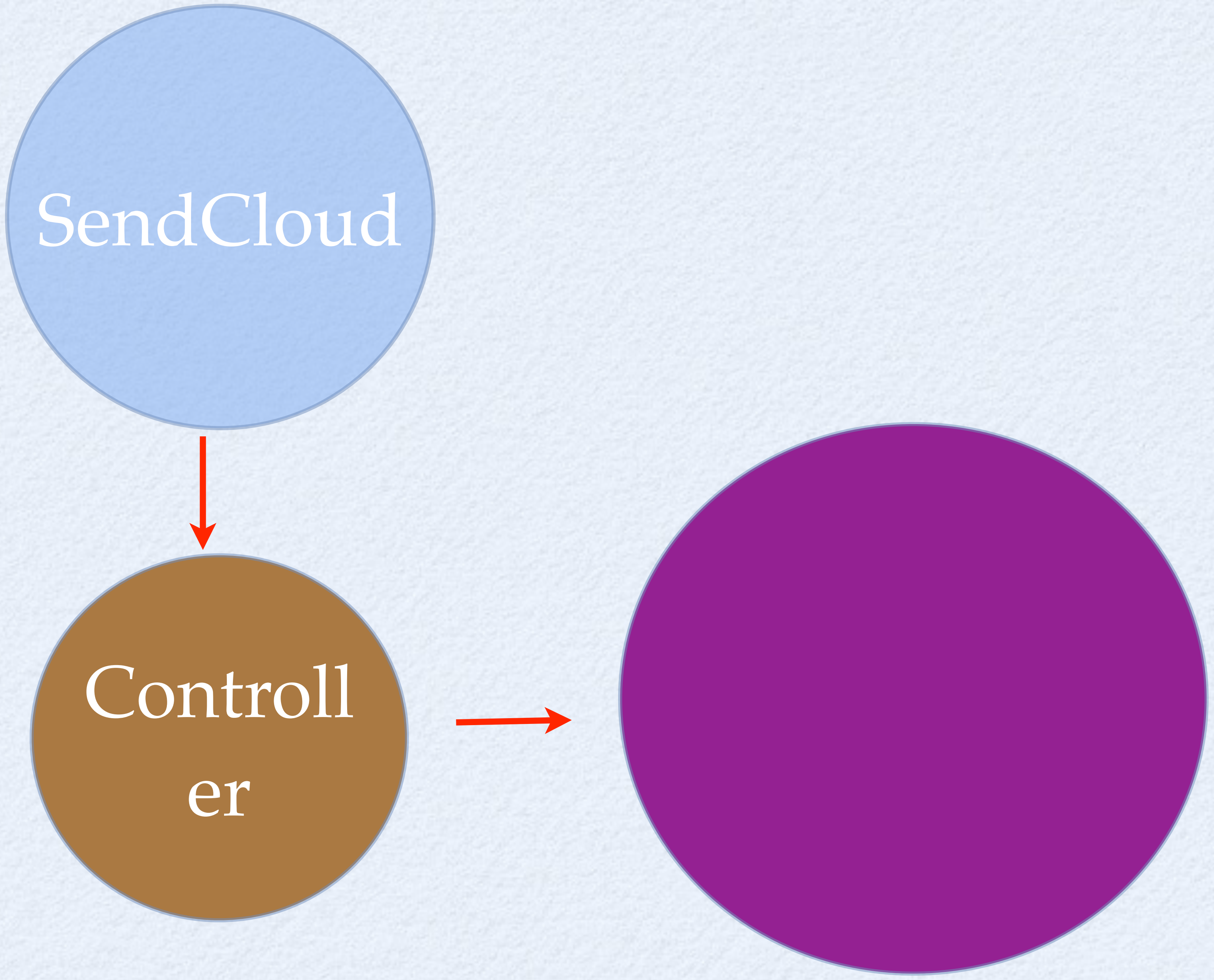
What we got

SendCloud
Controll
er

What we got

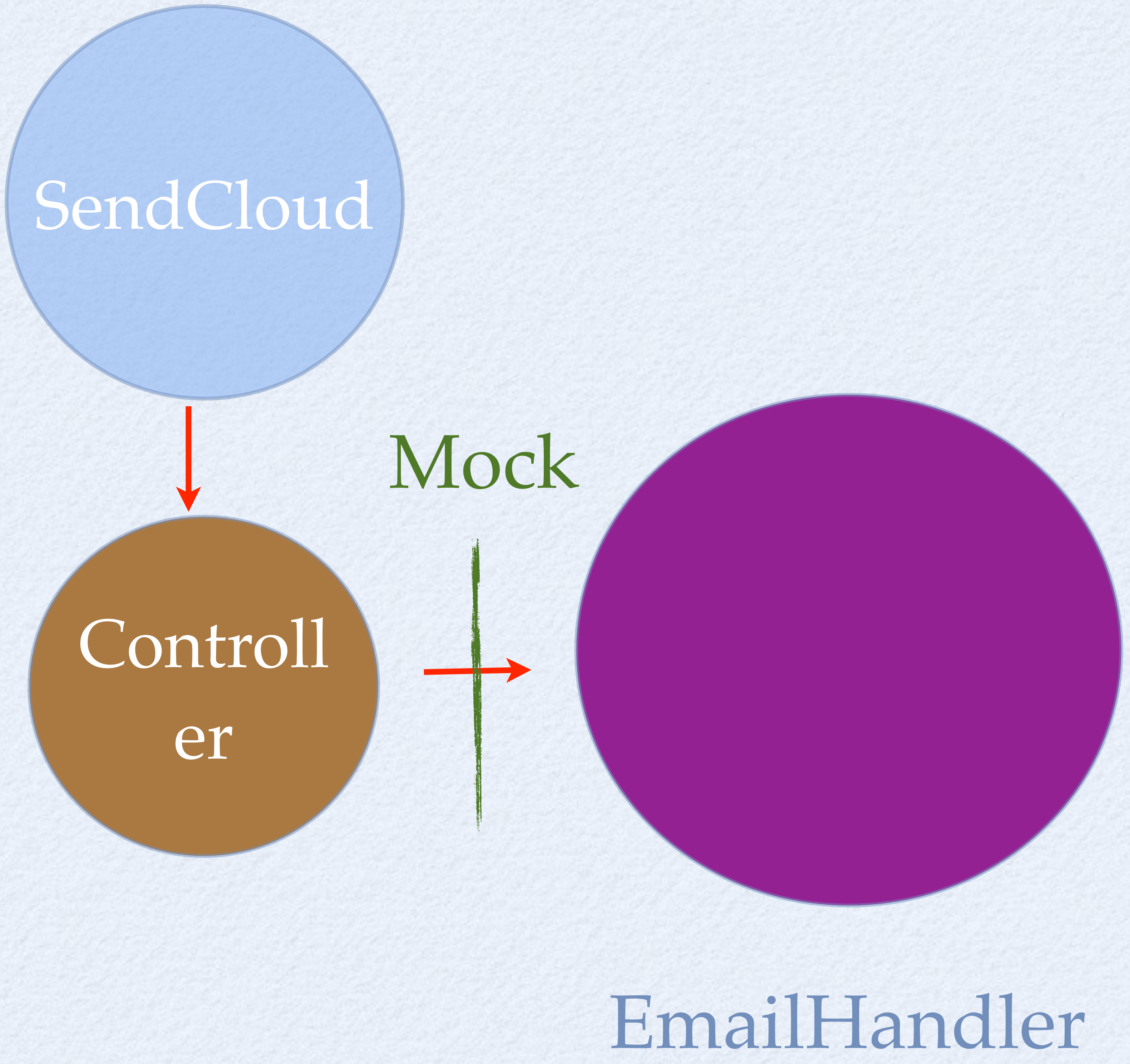


What we got

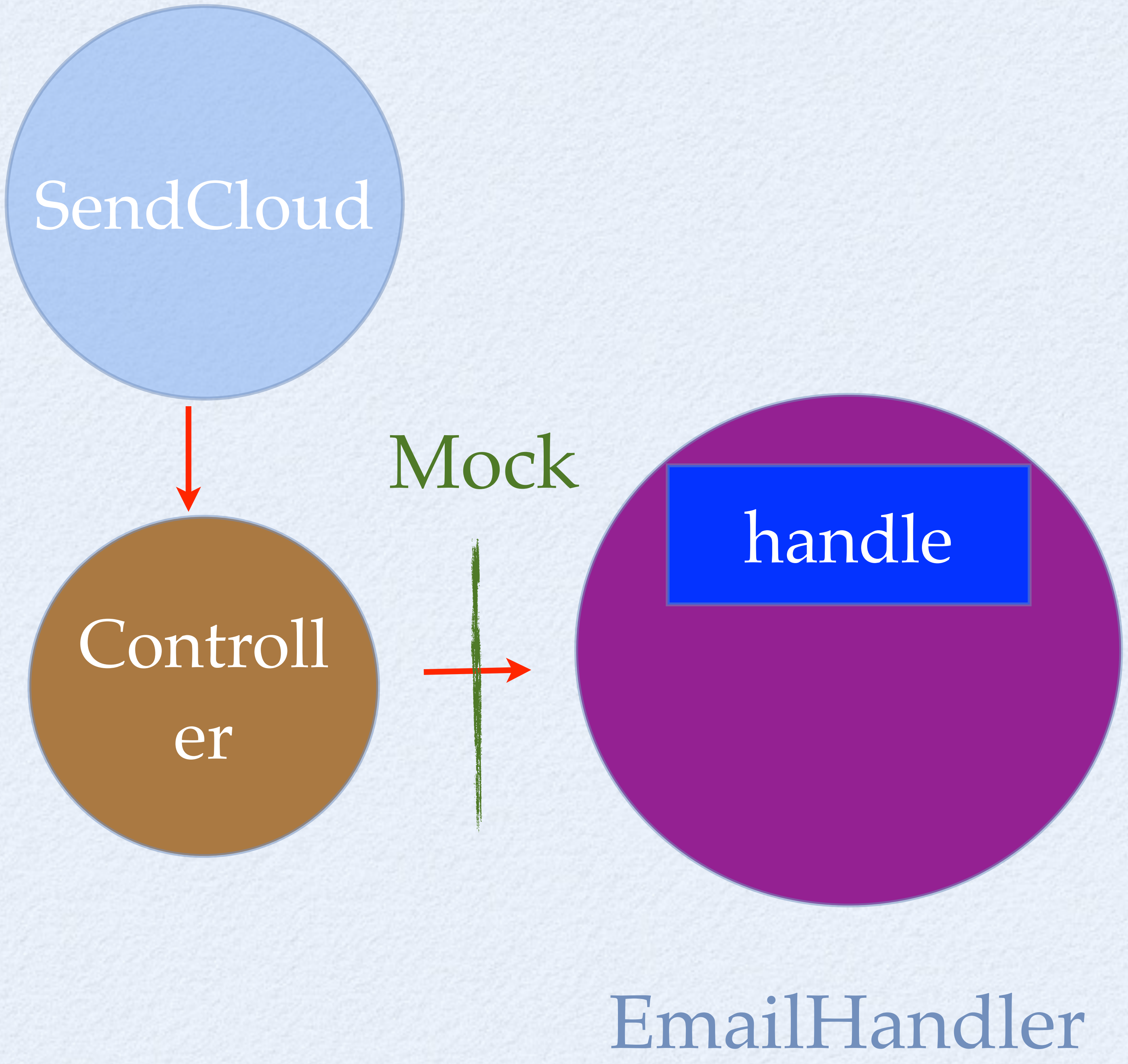


EmailHandler

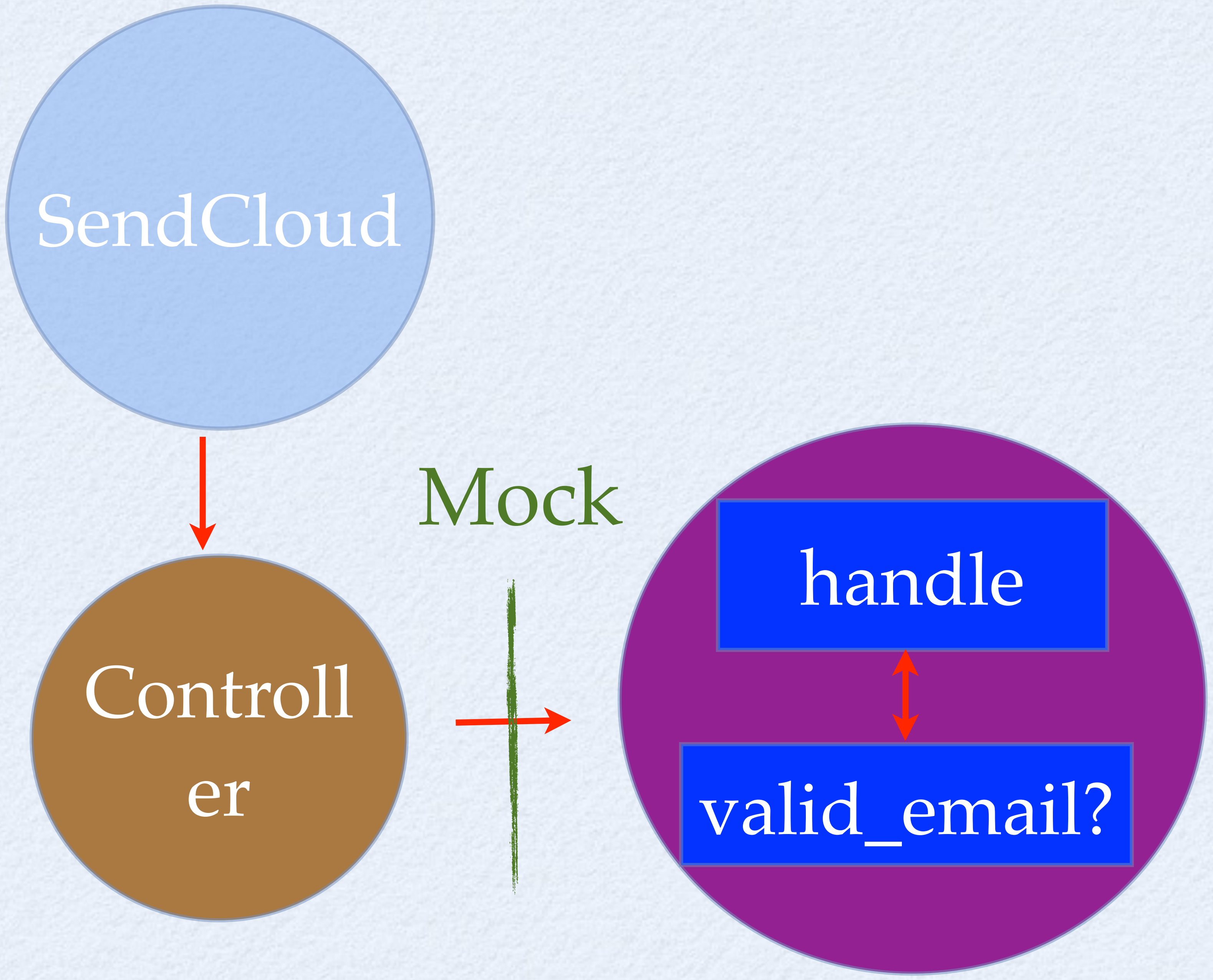
What we got



What we got

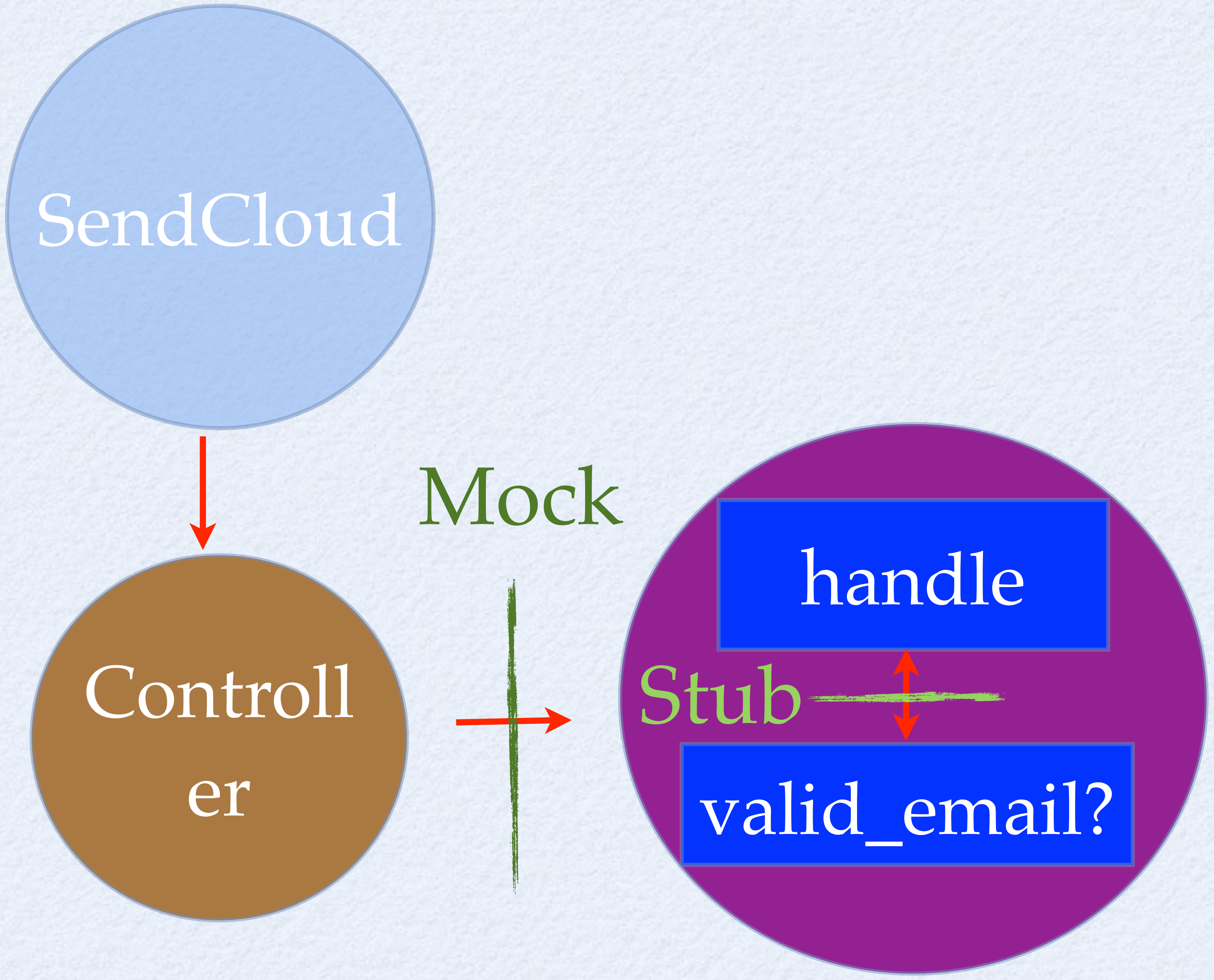


What we got



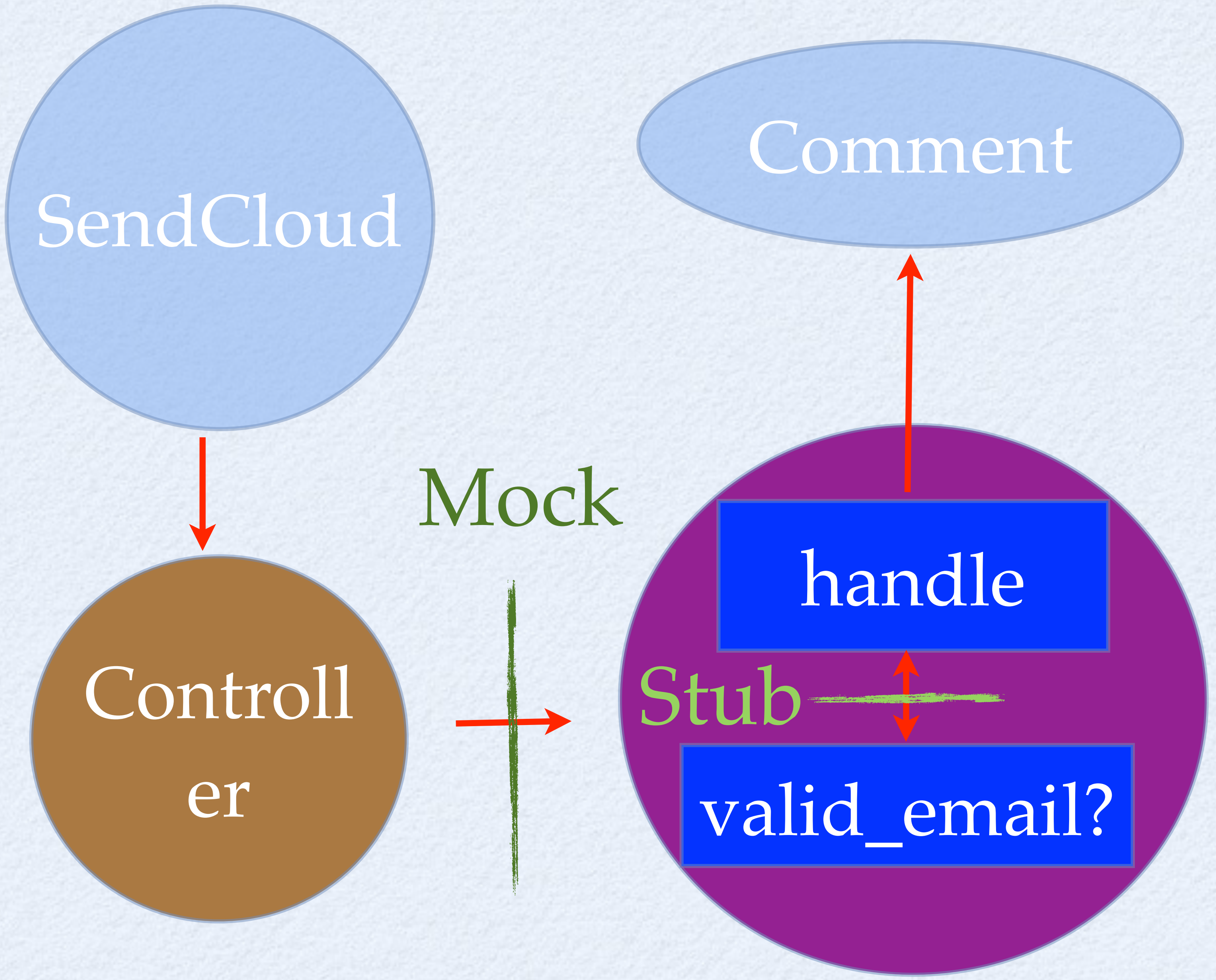
EmailHandler

What we got



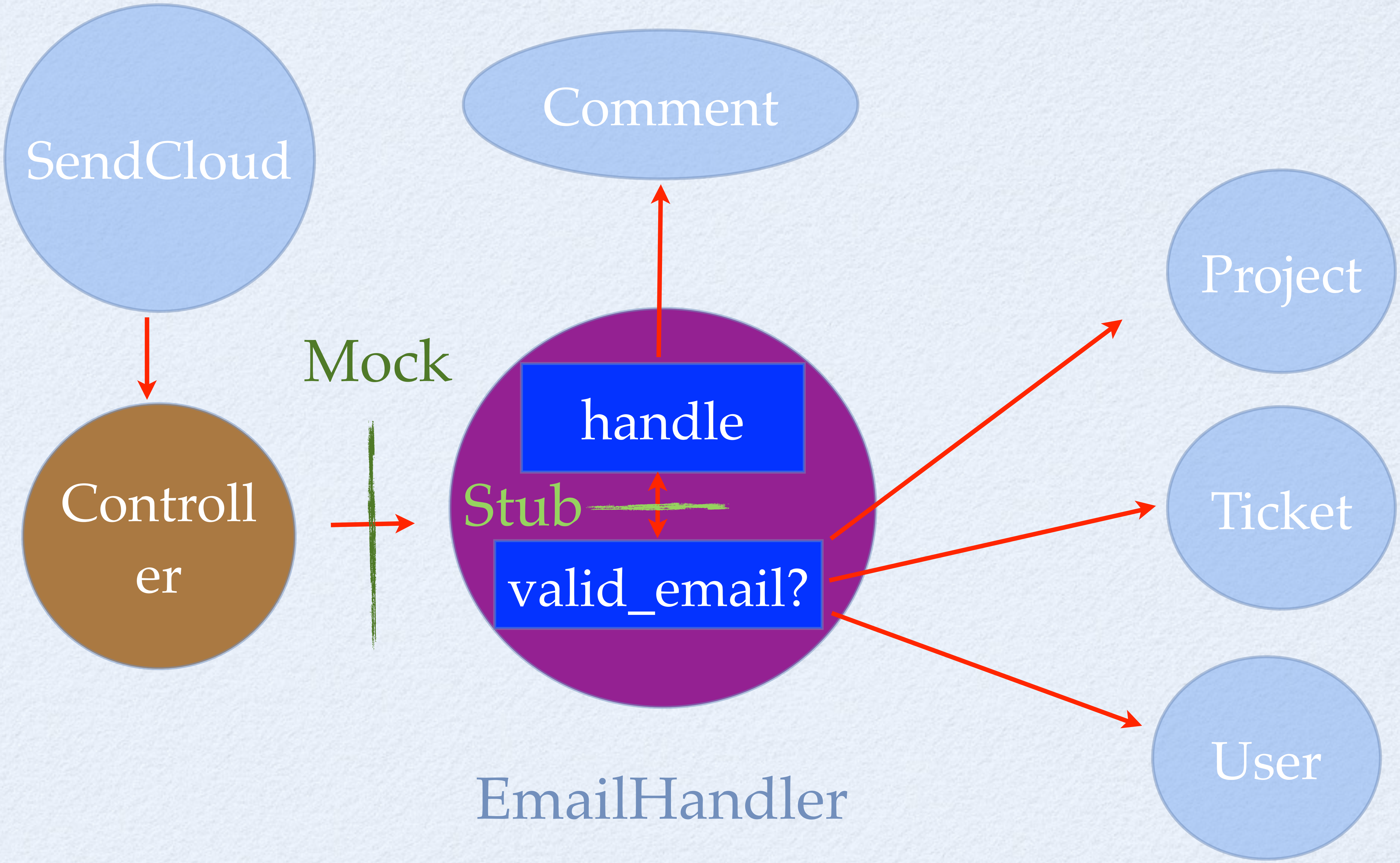
EmailHandler

What we got

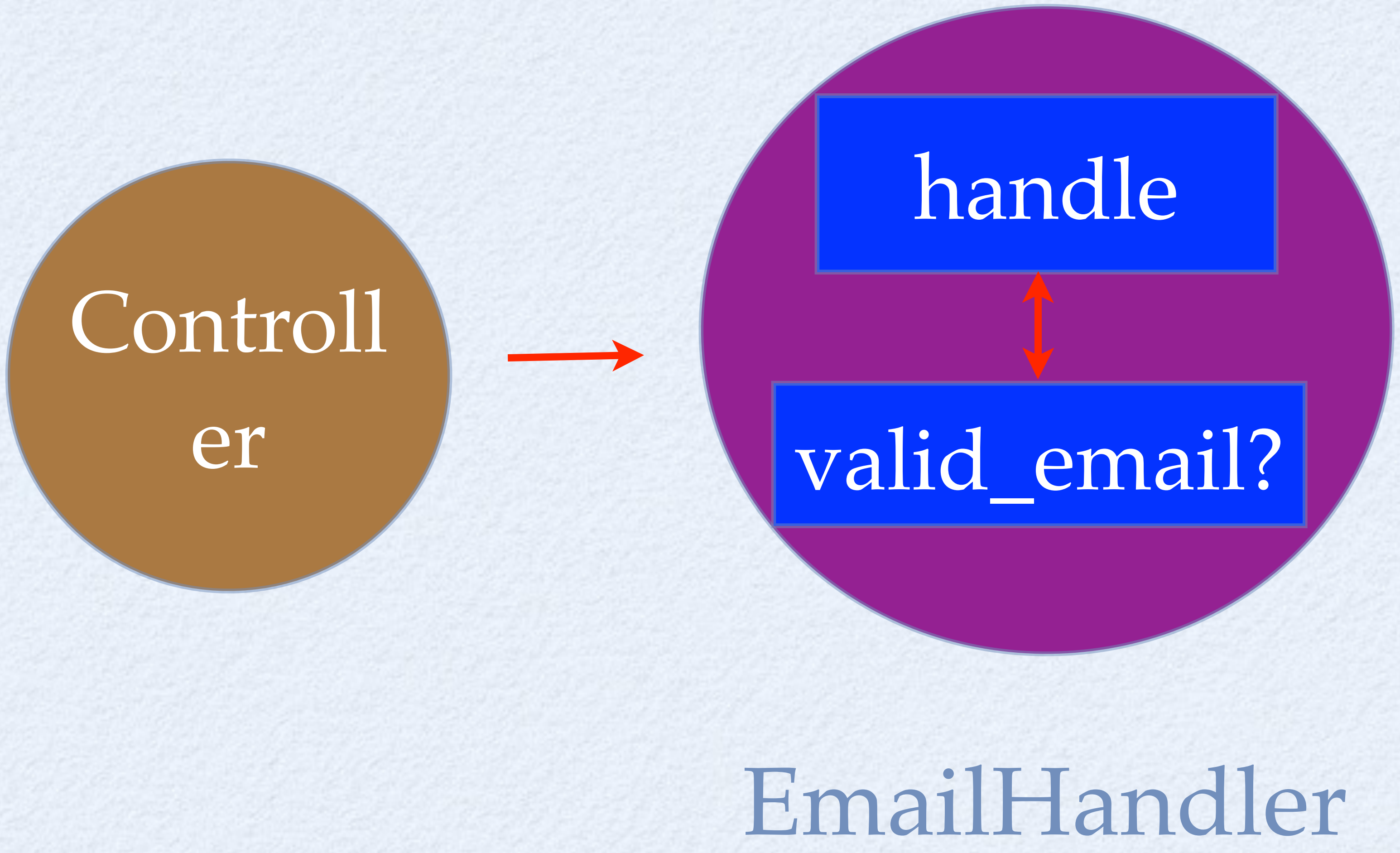


EmailHandler

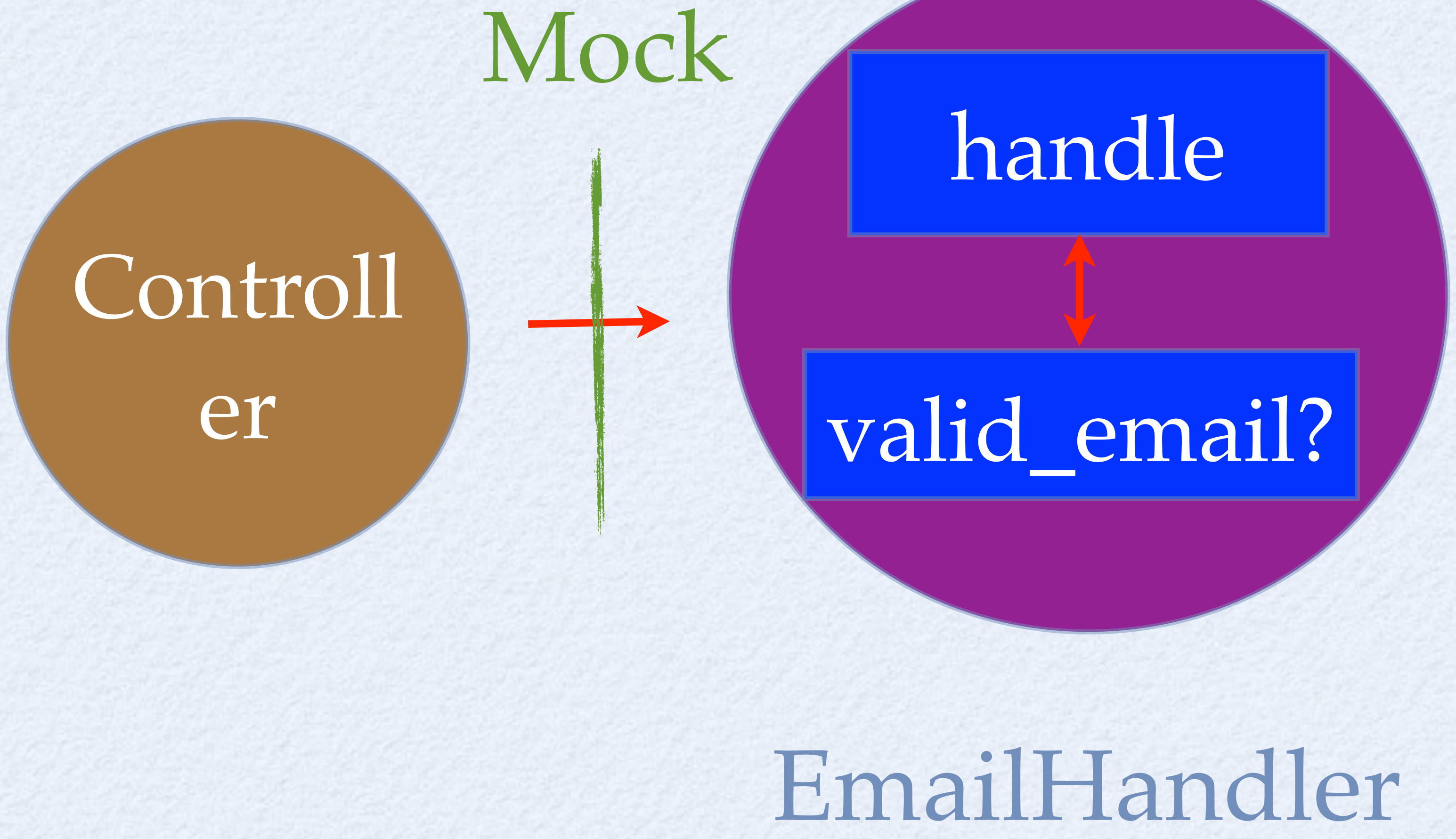
What we got


SendCloud
Comment
Project
Mock
Controll
er
handle
Stub
valid_email?
Ticket
User
EmailHandler

Mock and Stub

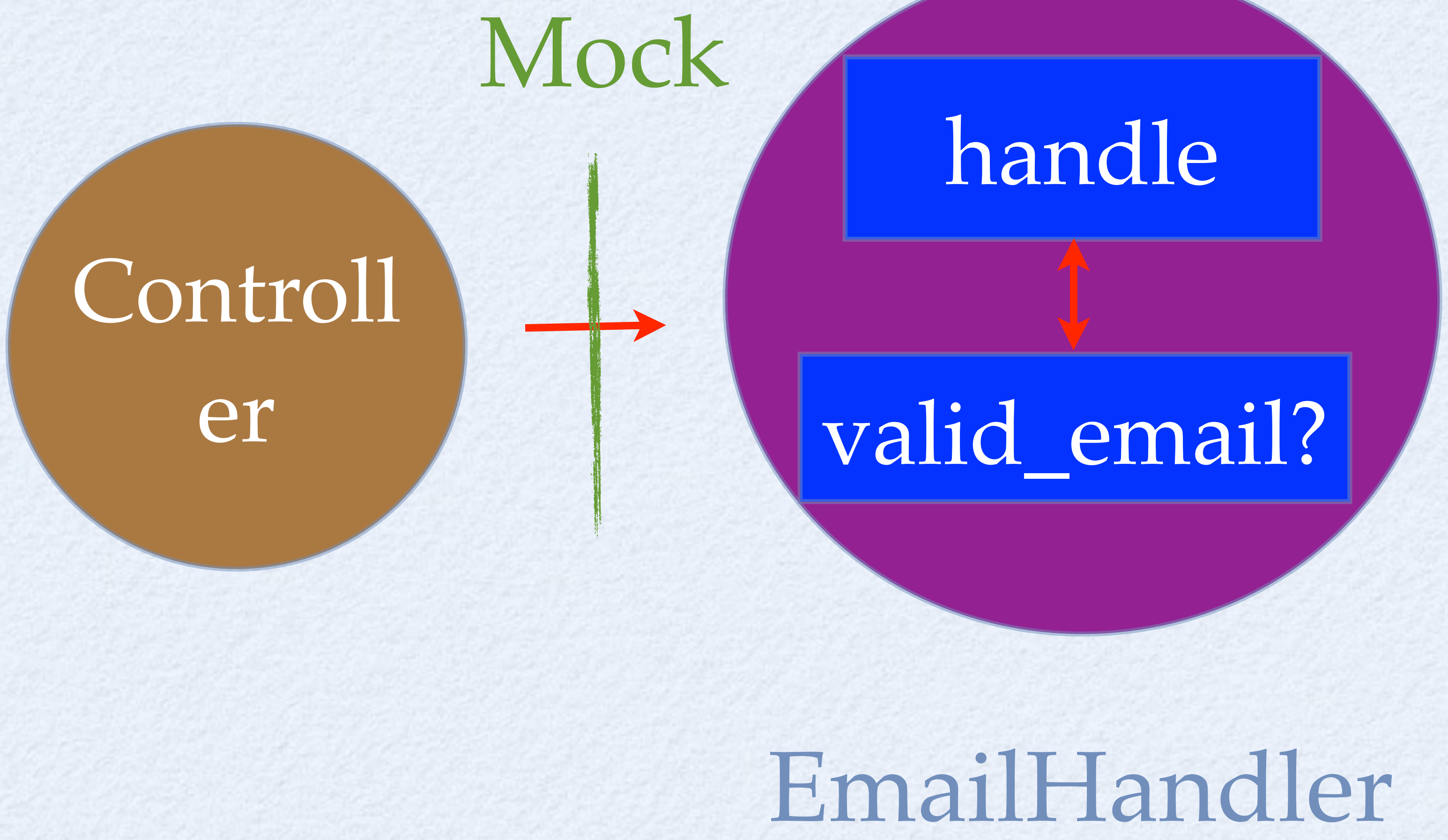


Mock and Stub



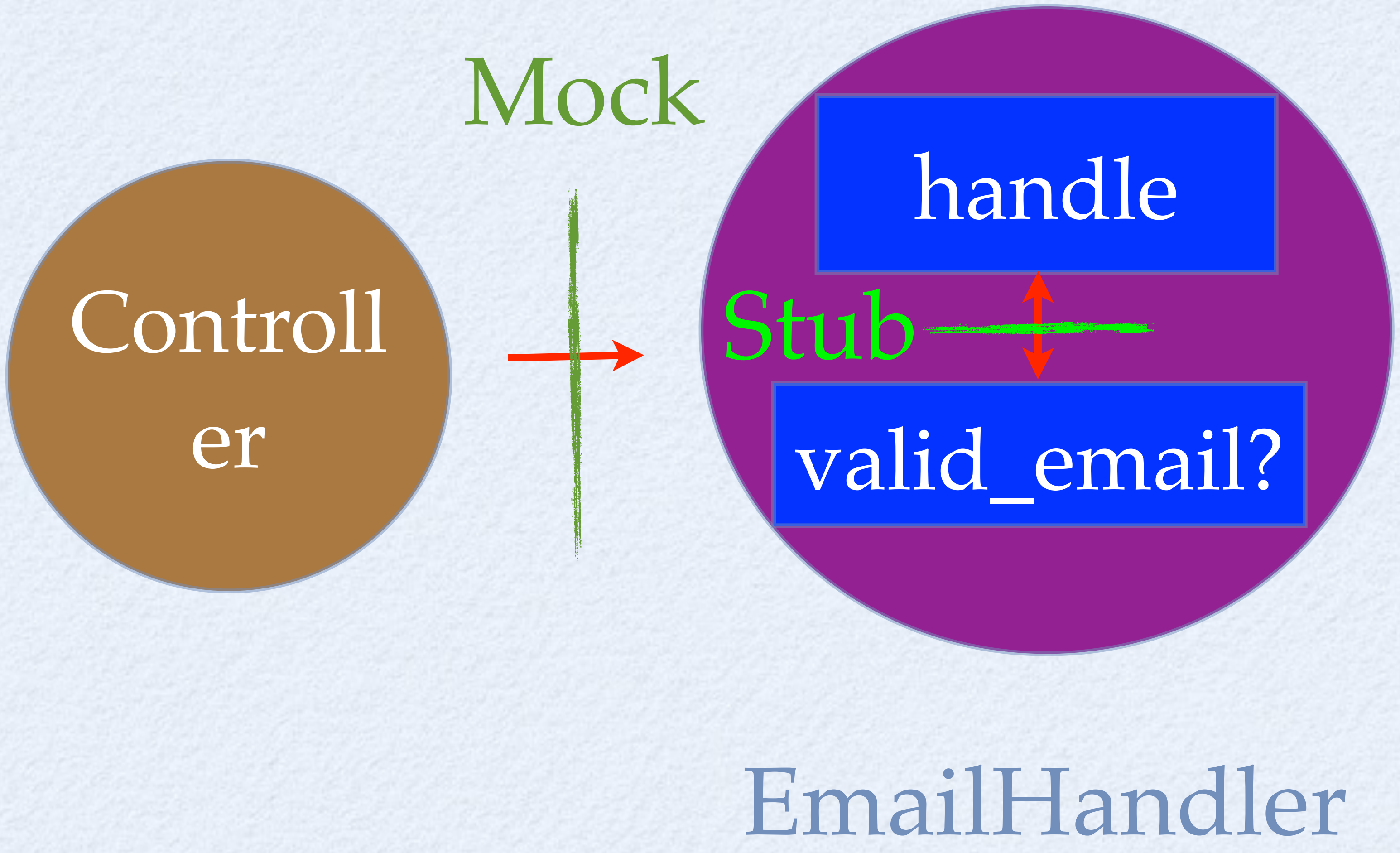
Mock and Stub

Mock: Command method.



Mock and Stub

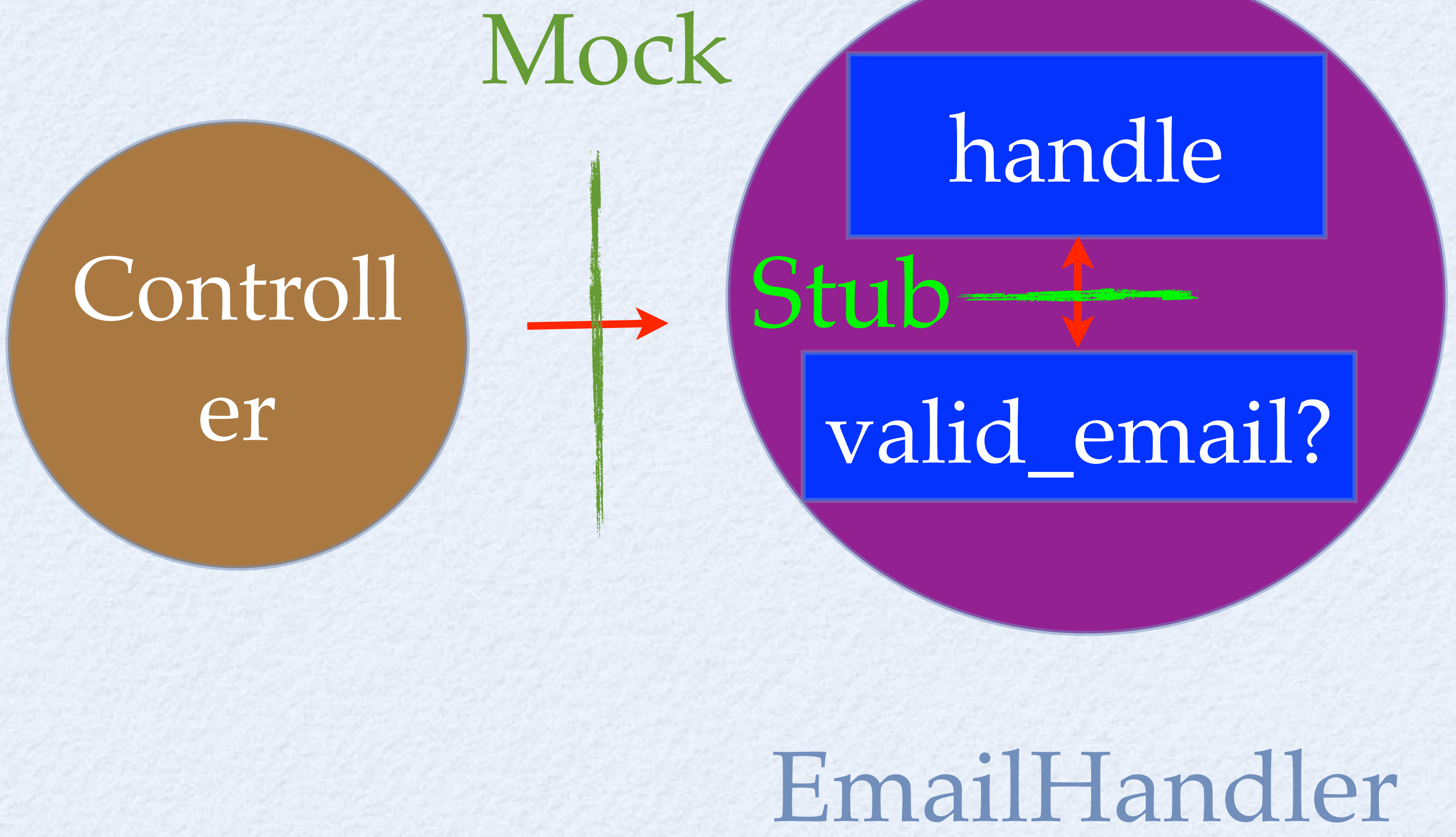
Mock: Command method.



Mock and Stub

Mock: Command method.

Stub: Query method.

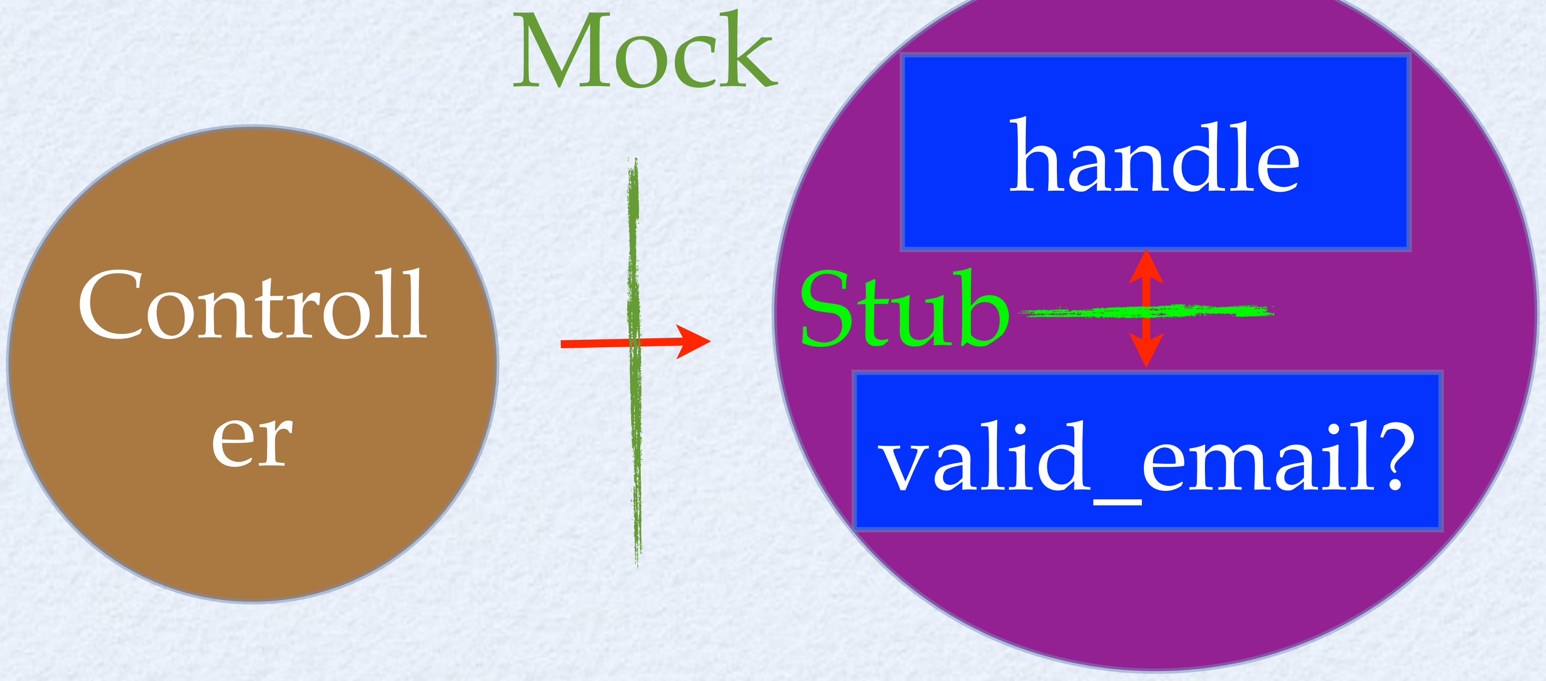


Mock and Stub

Mock: Command method.

Stub: Query method.

Mock is brittle.



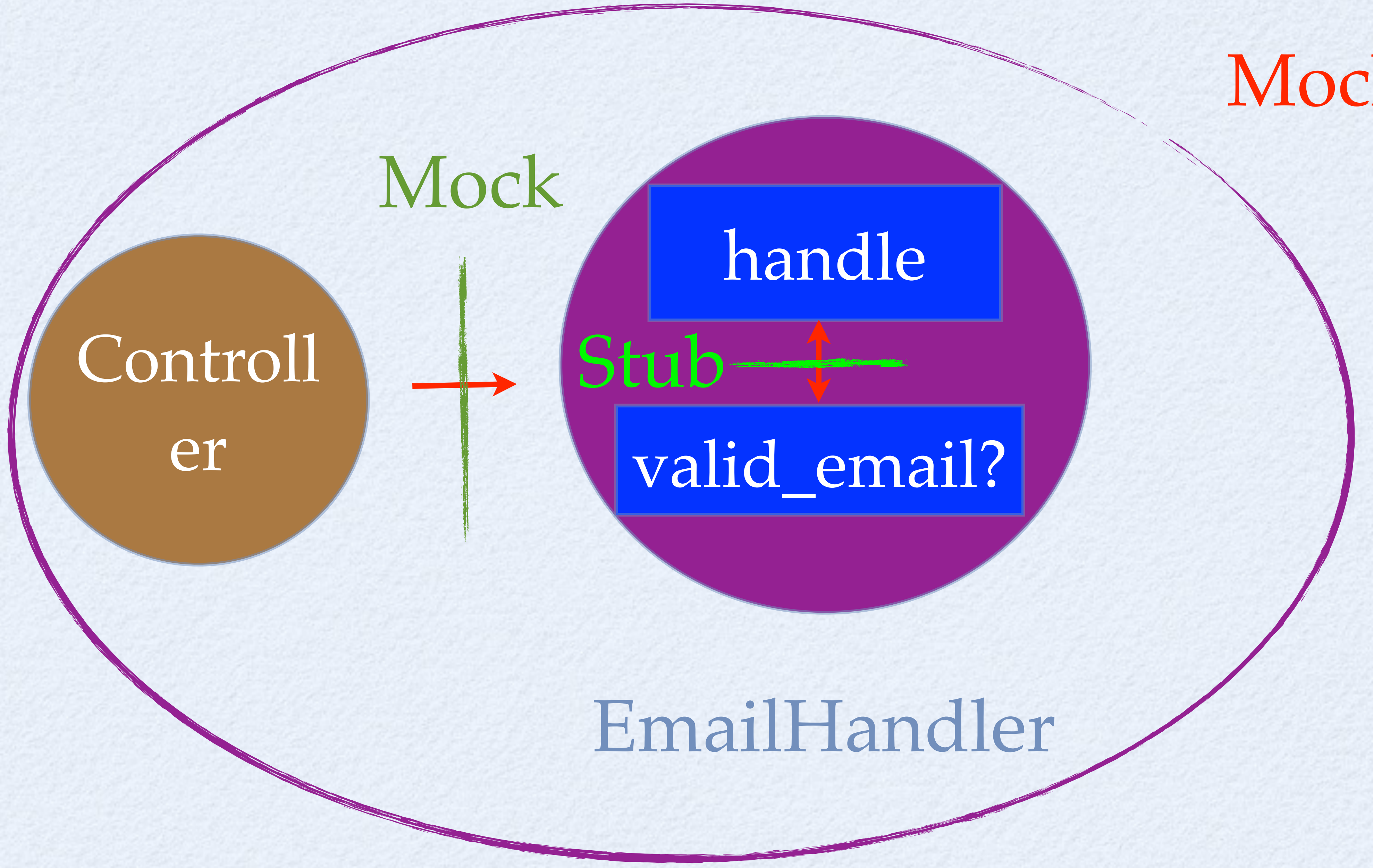
EmailHandler

Mock and Stub

Mock: Command method.

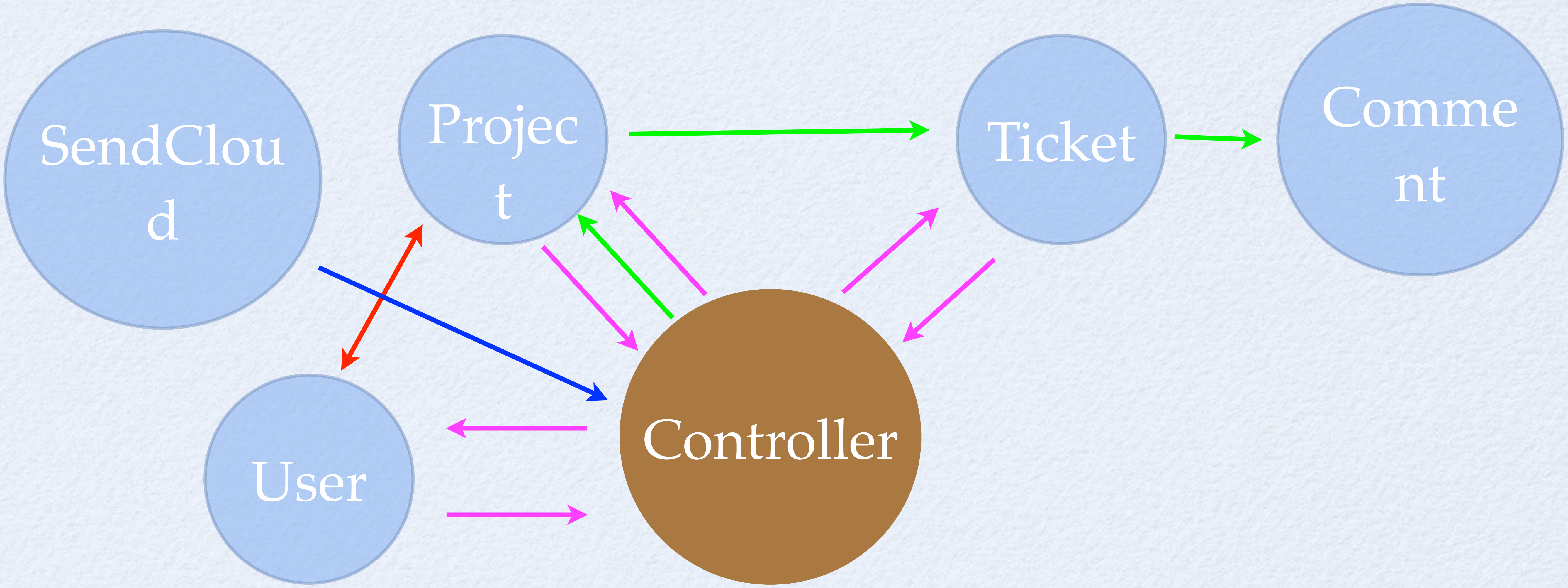
Stub: Query method.

Mock is brittle.

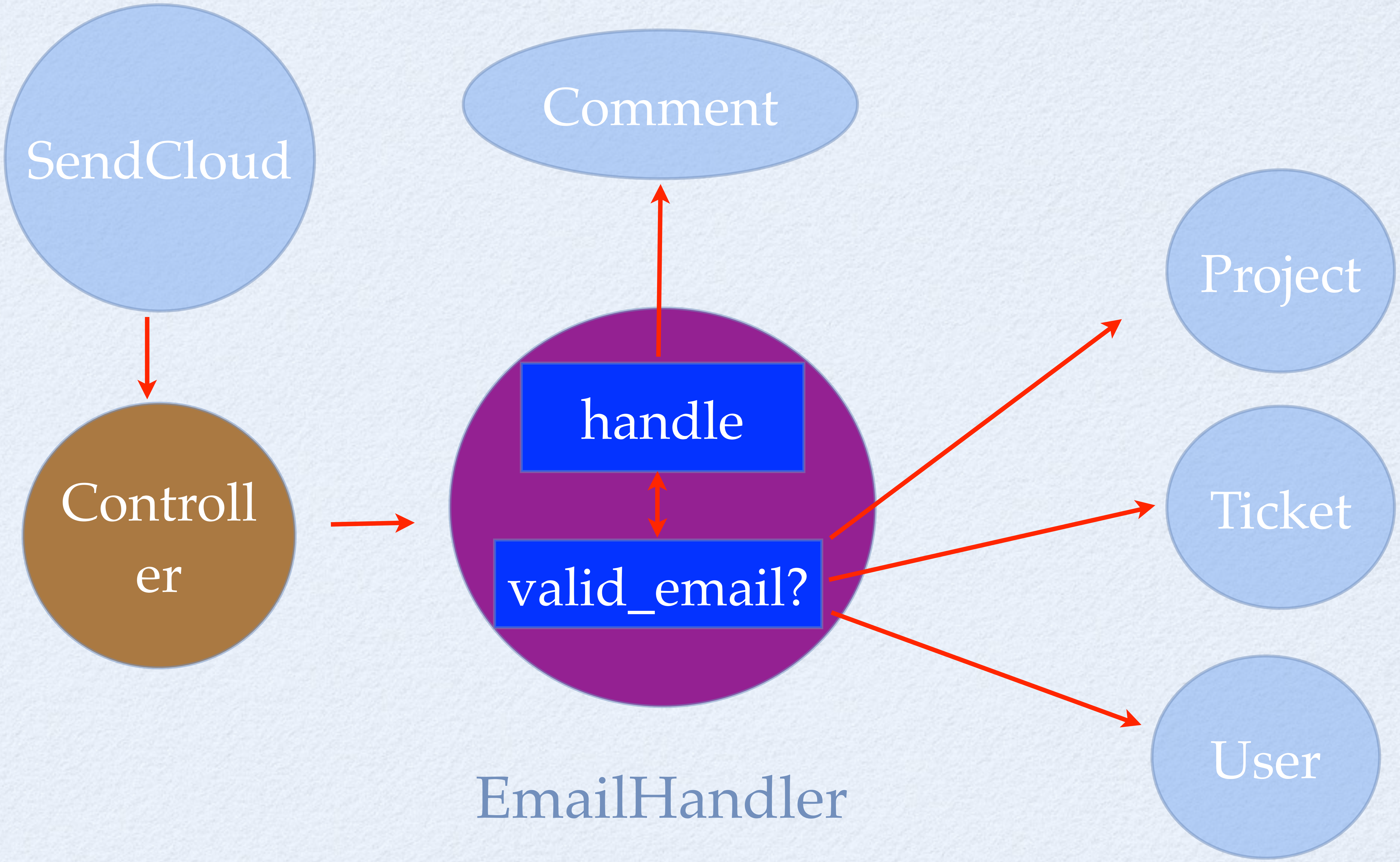


Integrate
Test

Before

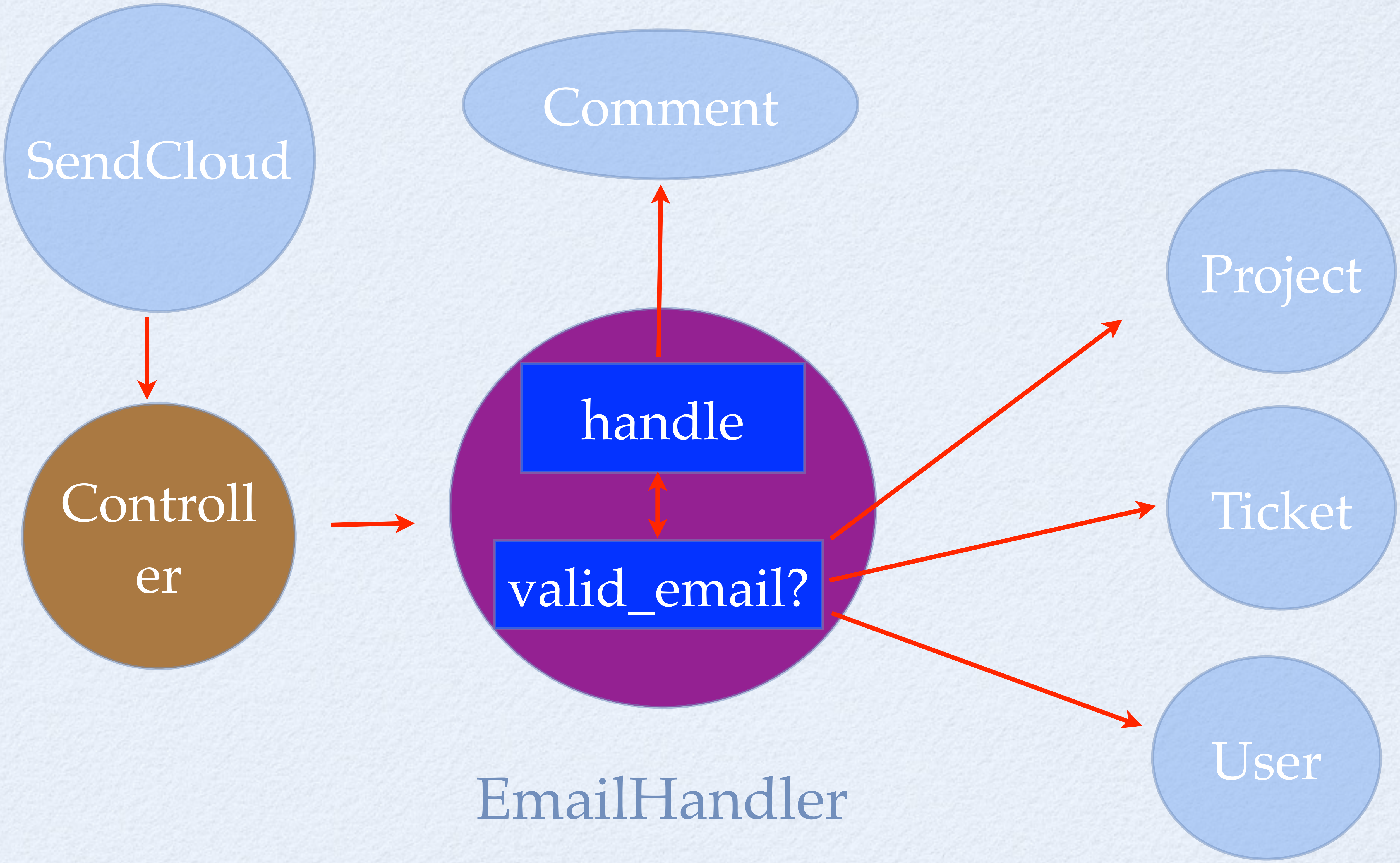


Benefits


SendCloud
Comment
Project
Controll
er
handle
valid_email?
Ticket
User
EmailHandler

Benefits

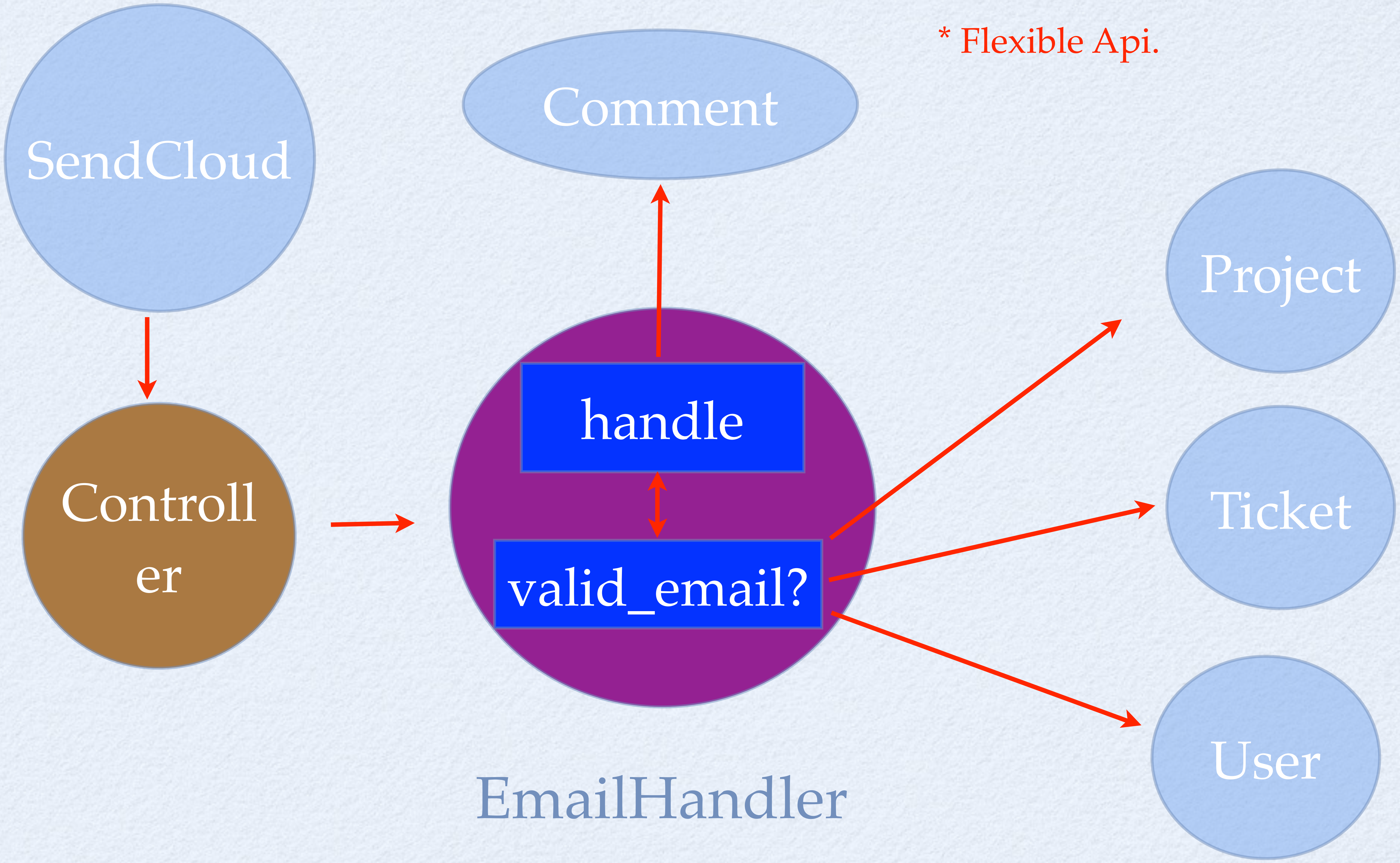
* Test could be a documentation.



Benefits

* Test could be a documentation.

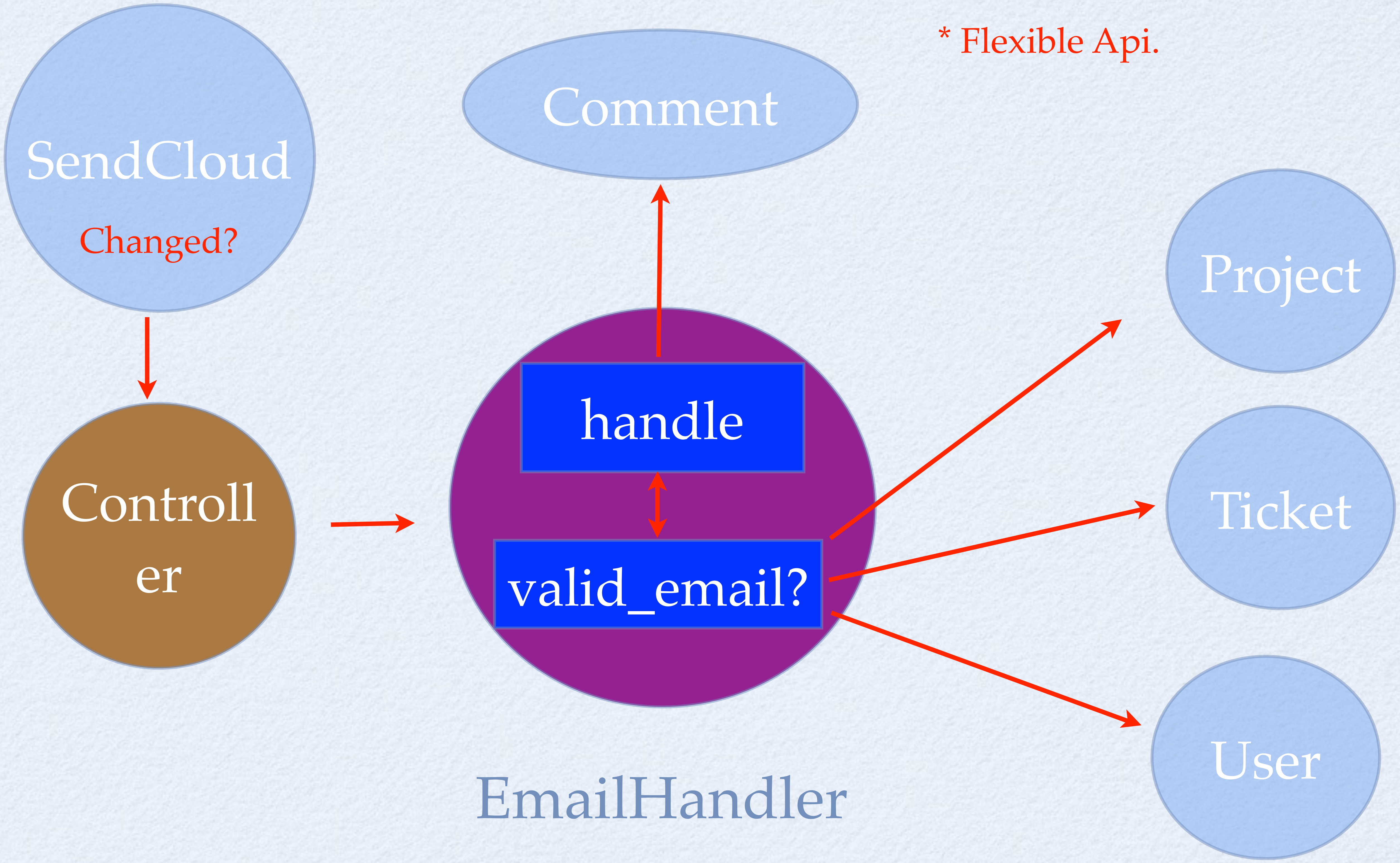
* Flexible Api.



Benefits

* Test could be a documentation.

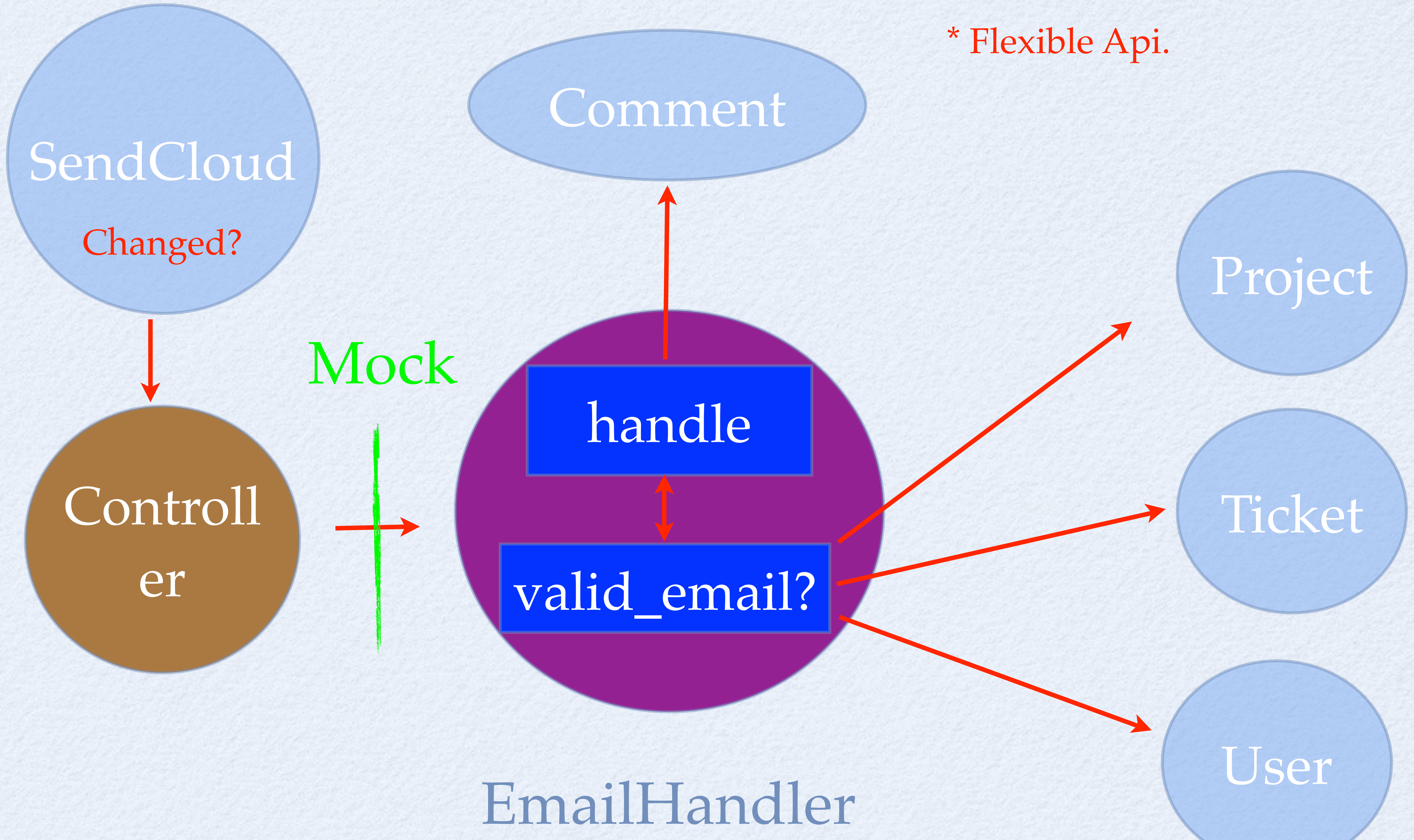
* Flexible Api.



Benefits

* Test could be a documentation.

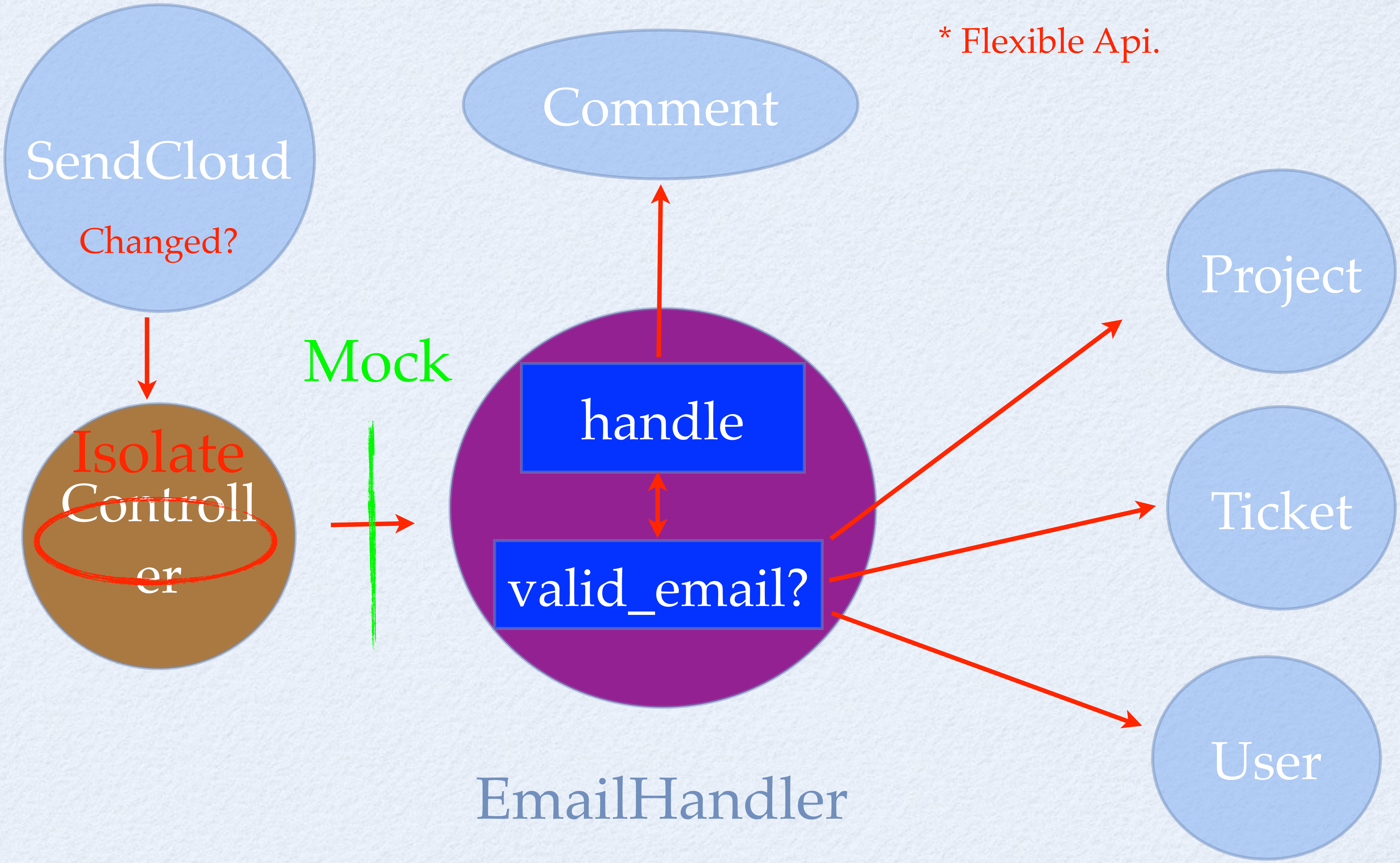
* Flexible Api.



Benefits

* Test could be a documentation.

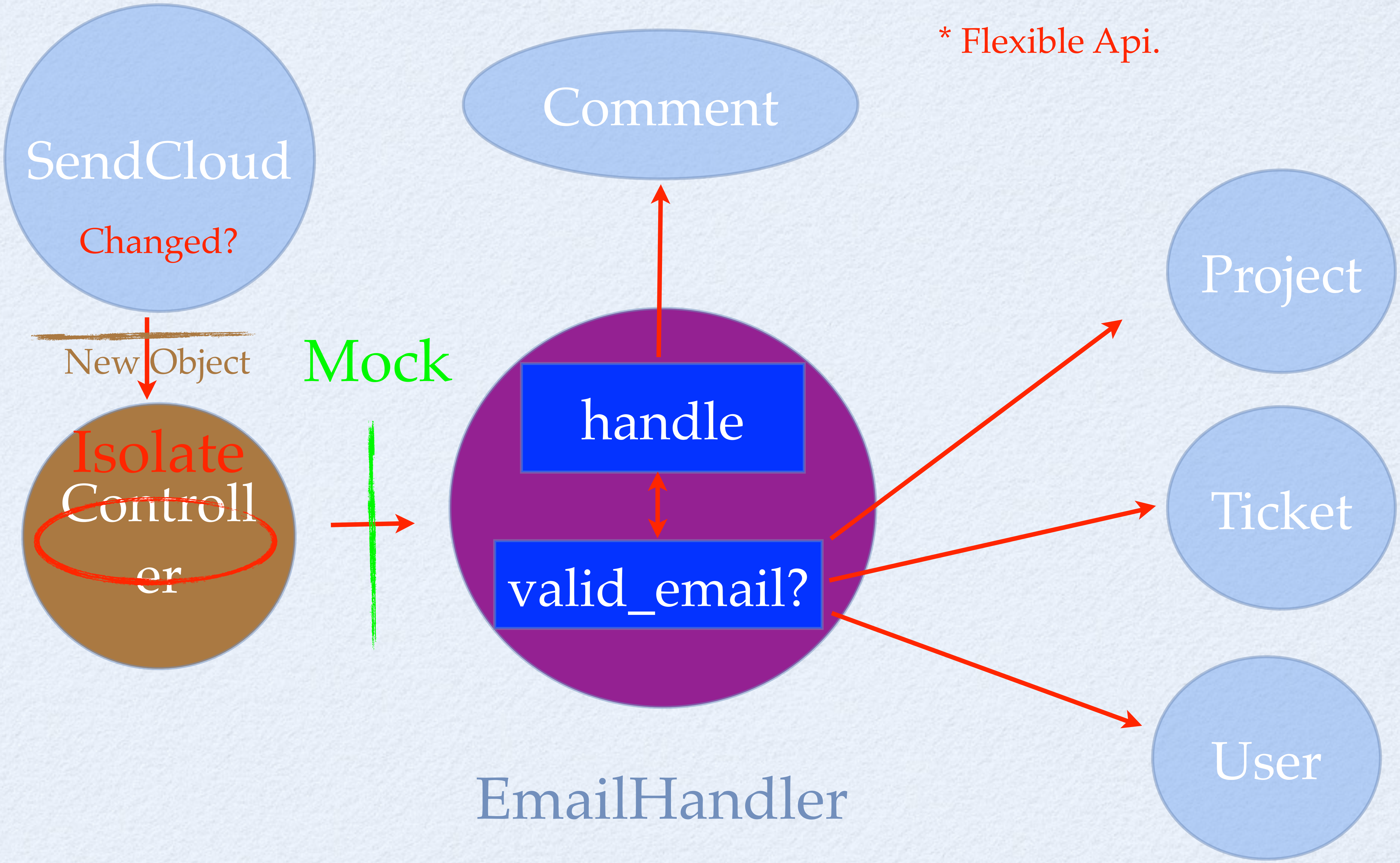
* Flexible Api.



Benefits

* Test could be a documentation.

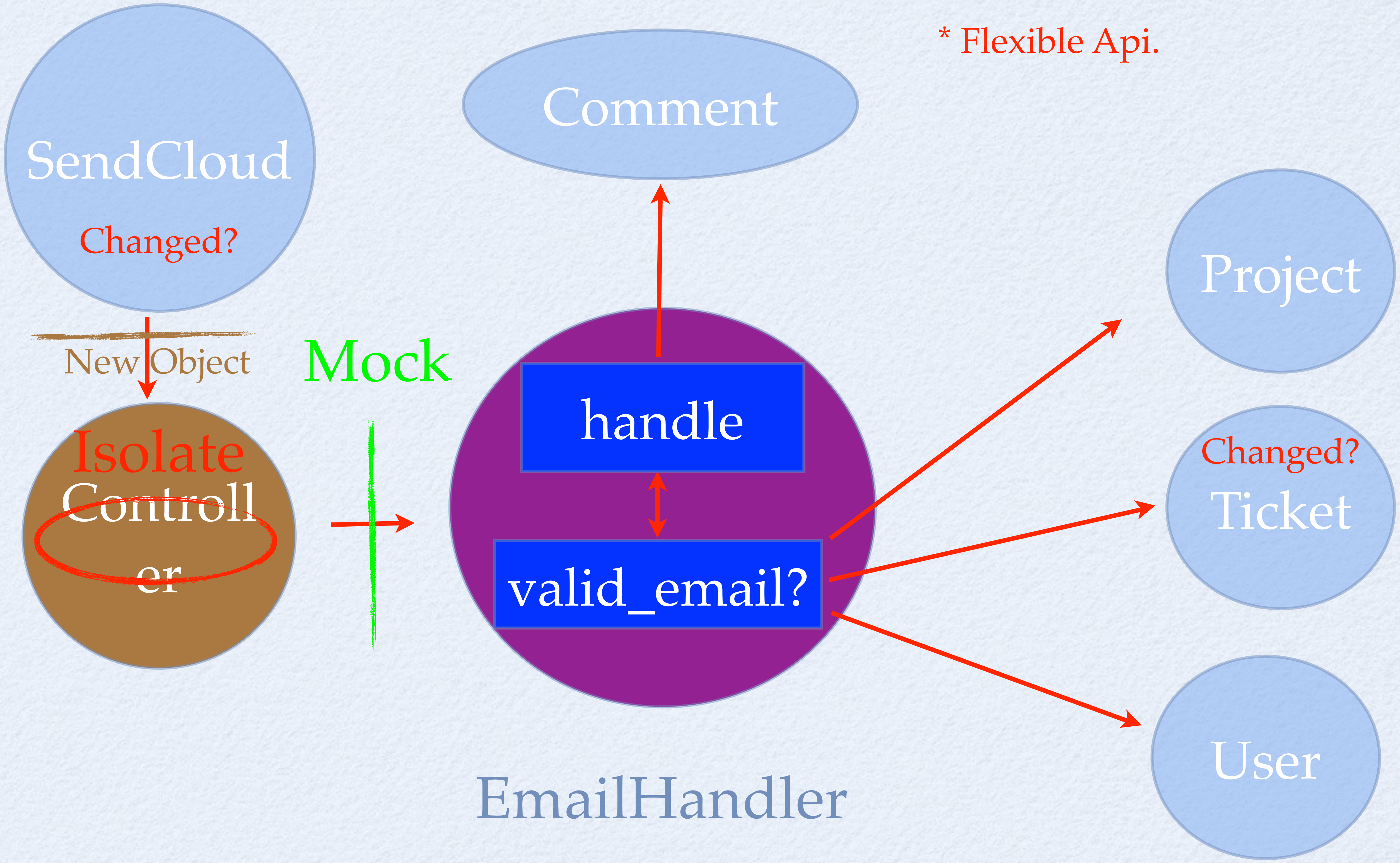
* Flexible Api.



Benefits

* Test could be a documentation.

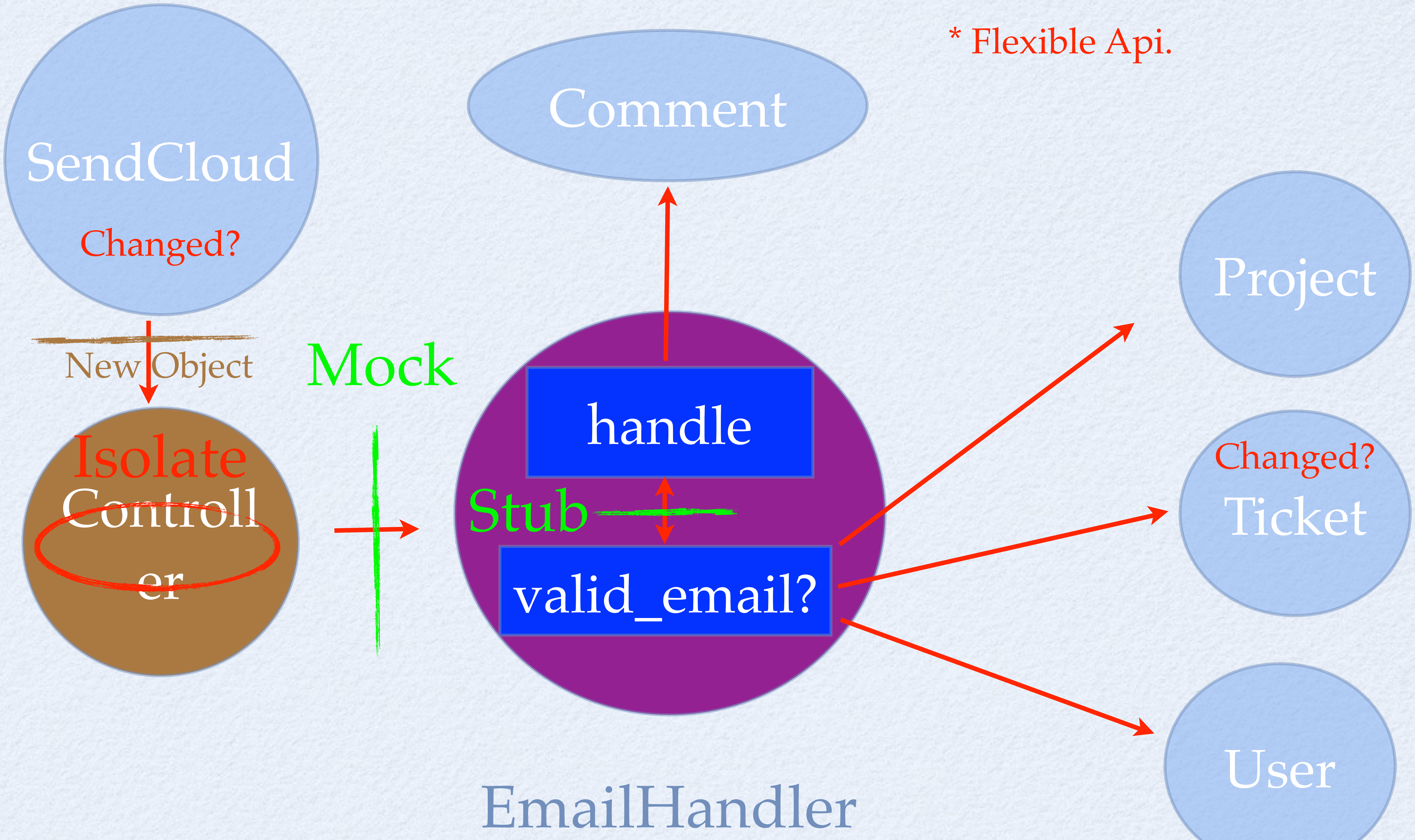
* Flexible Api.



Benefits

* Test could be a documentation.

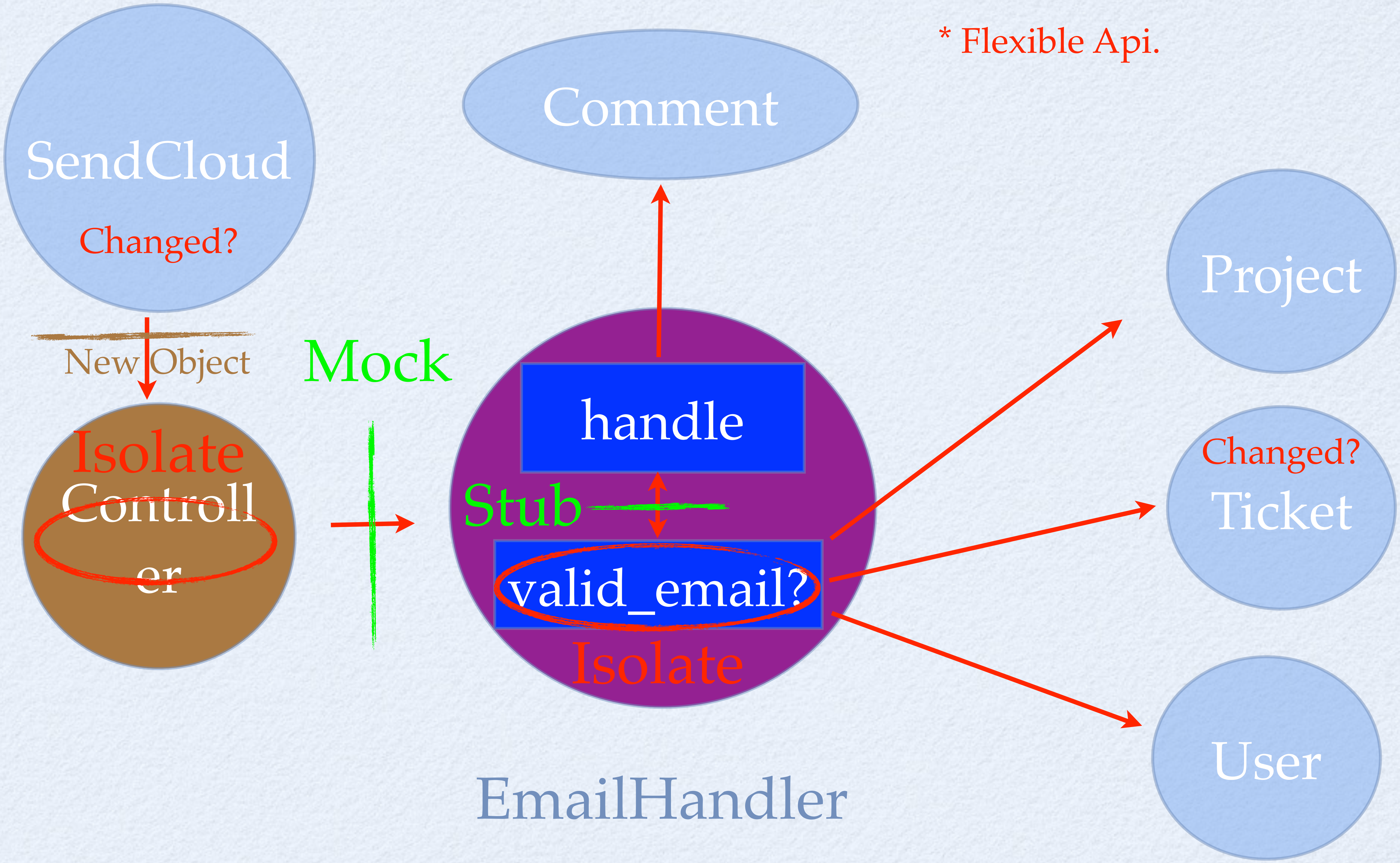
* Flexible Api.



Benefits

* Test could be a documentation.

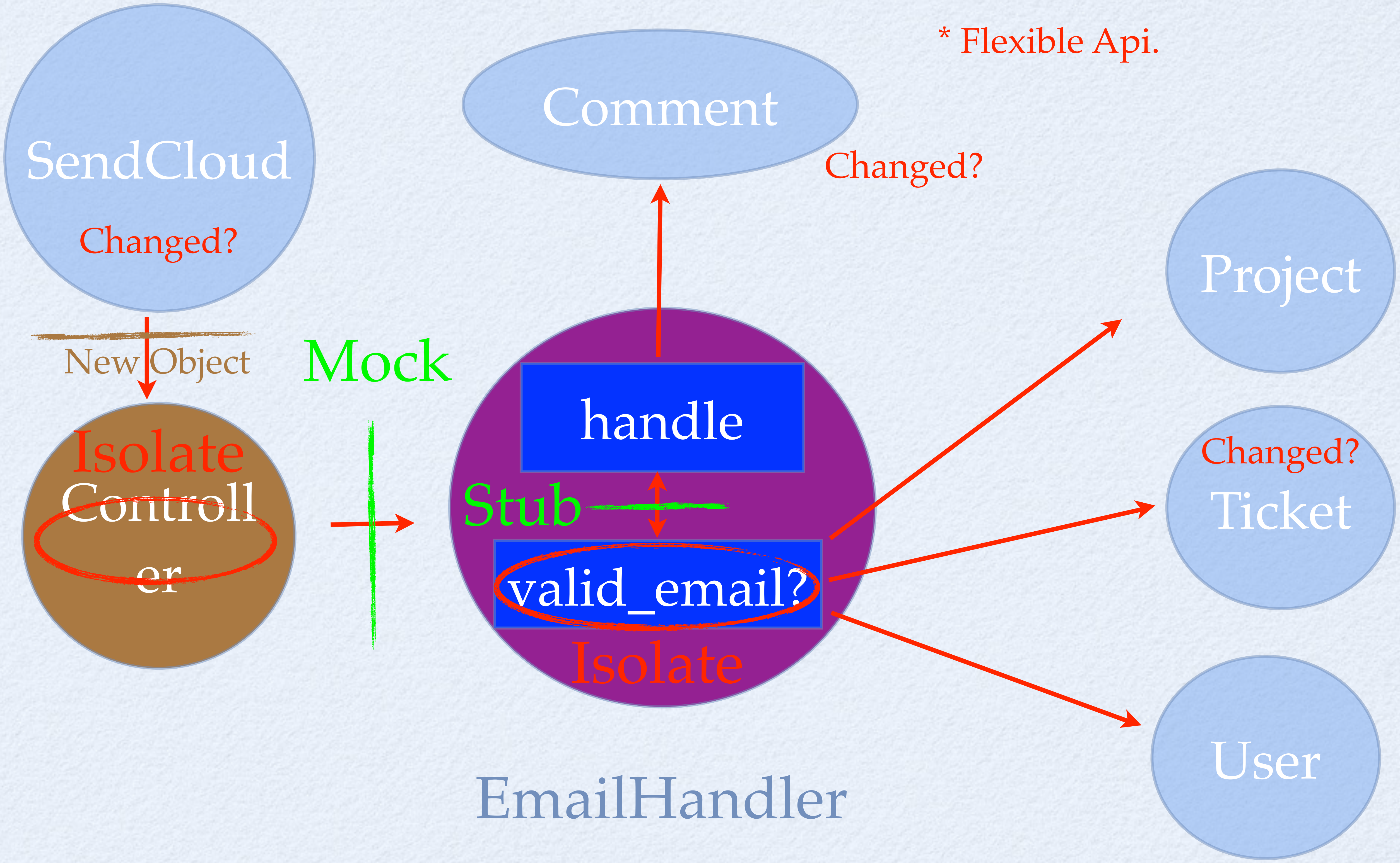
* Flexible Api.



Benefits

* Test could be a documentation.

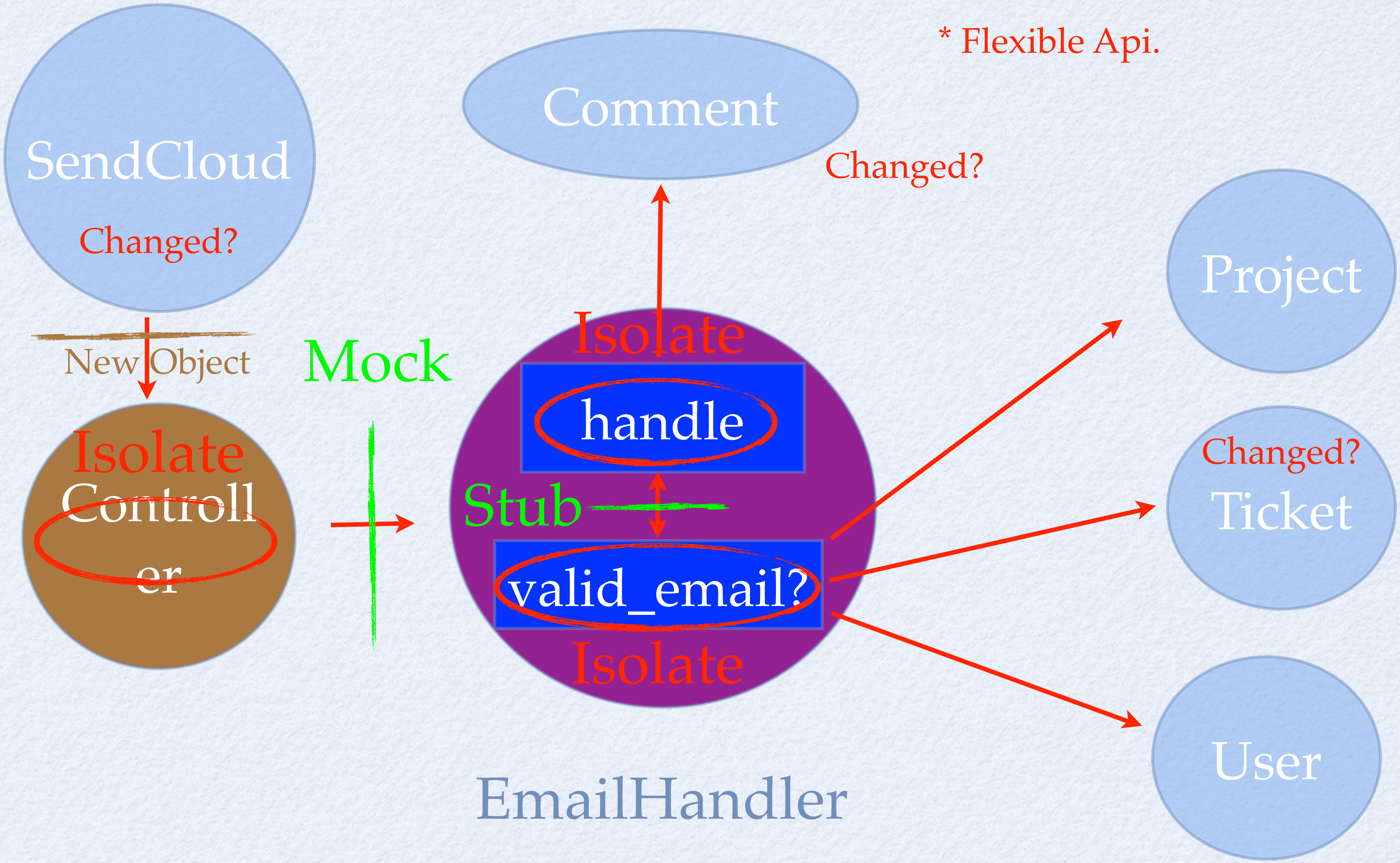
* Flexible Api.



Benefits

* Test could be a documentation.

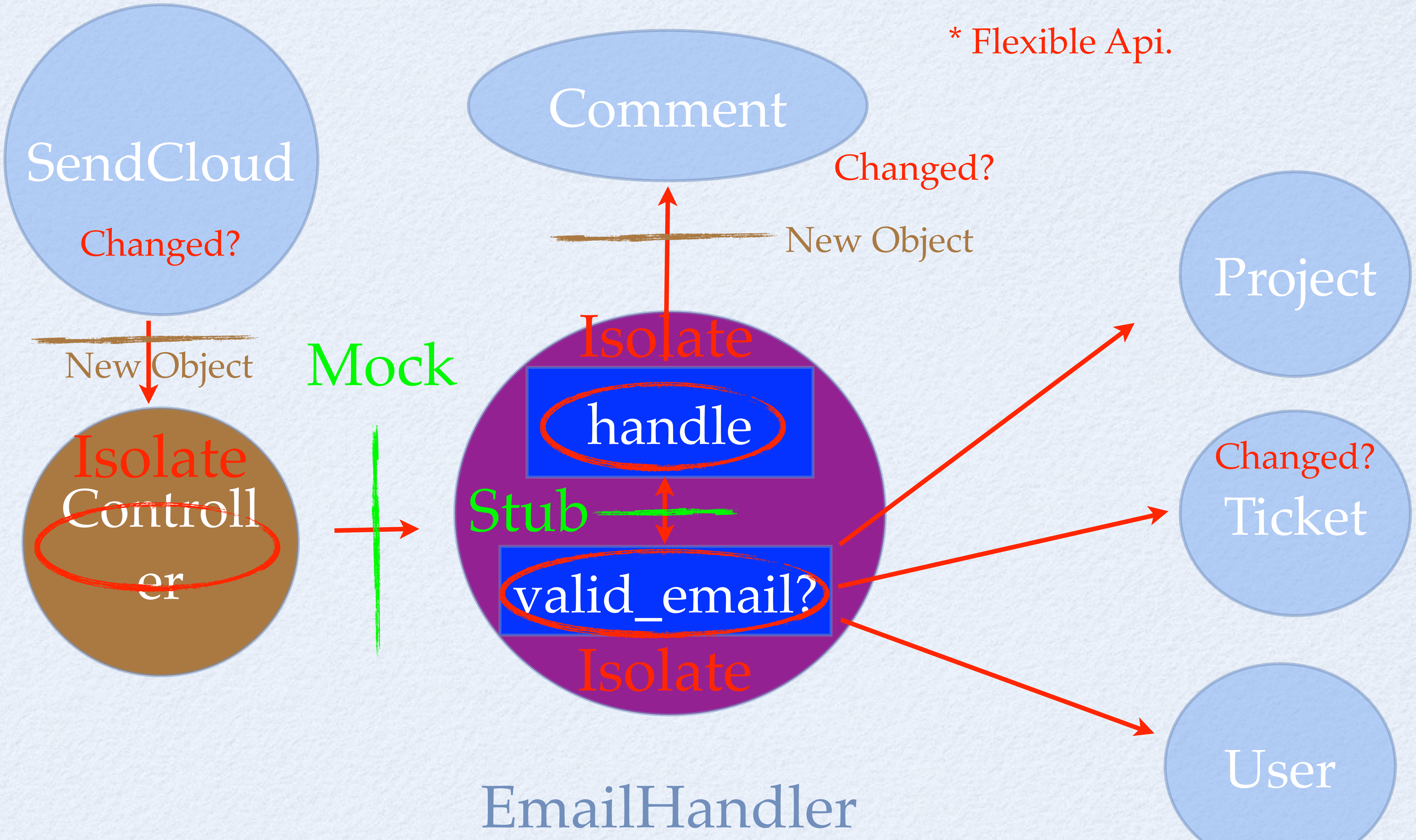
* Flexible Api.



Benefits

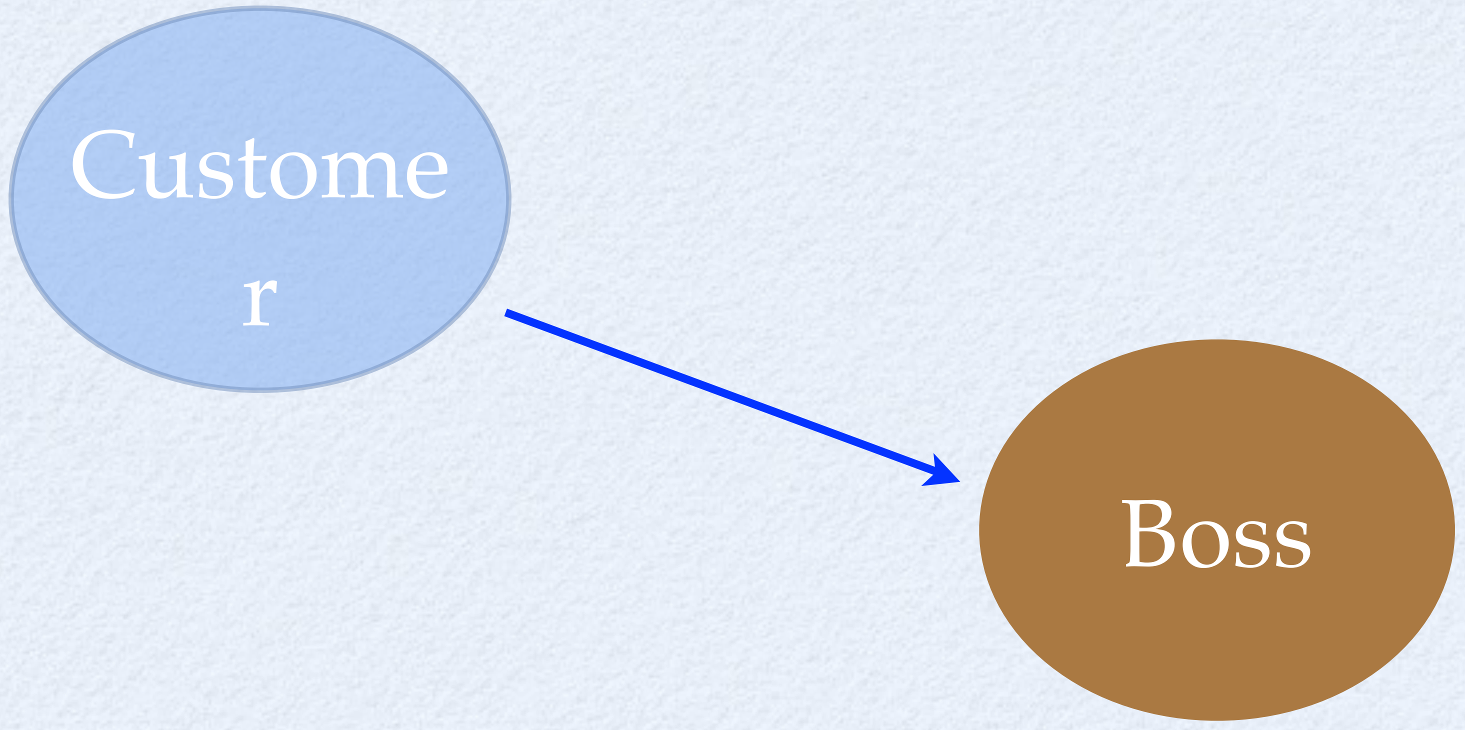
* Test could be a documentation.

* Flexible Api.

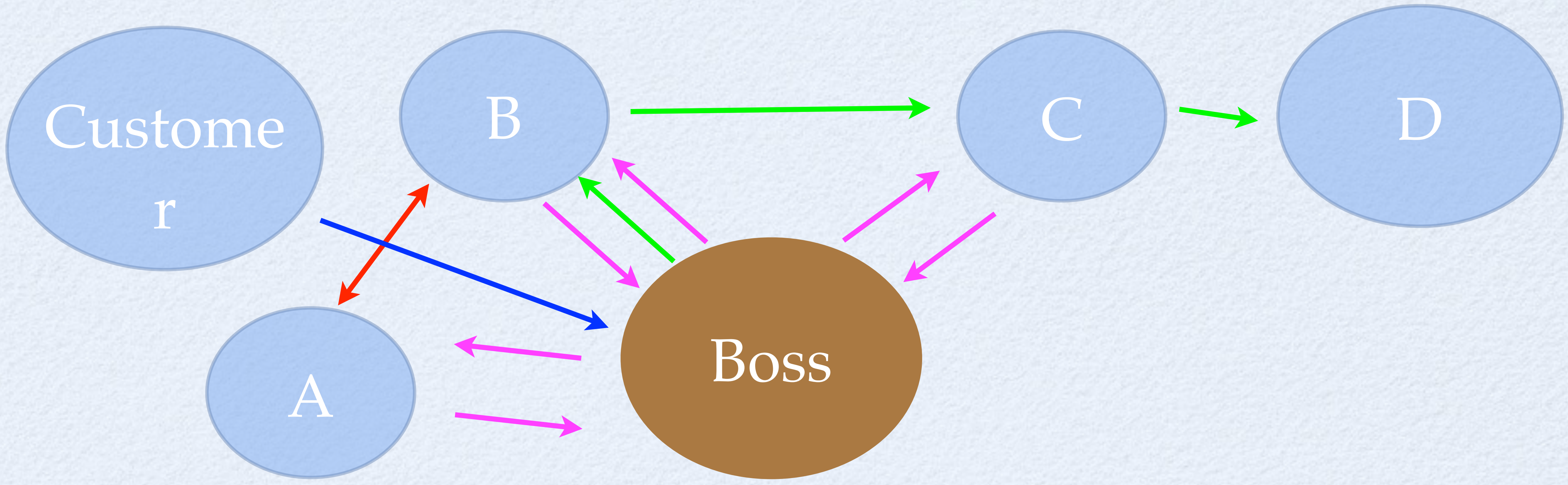


If you don’t get me…

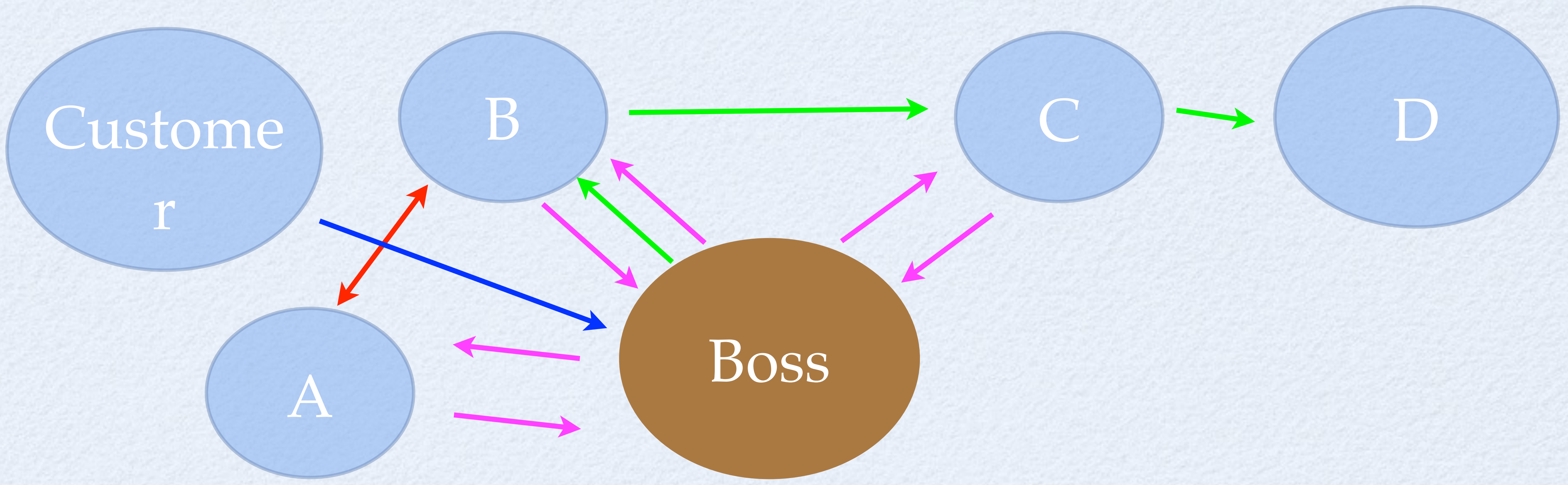
Example & Principles

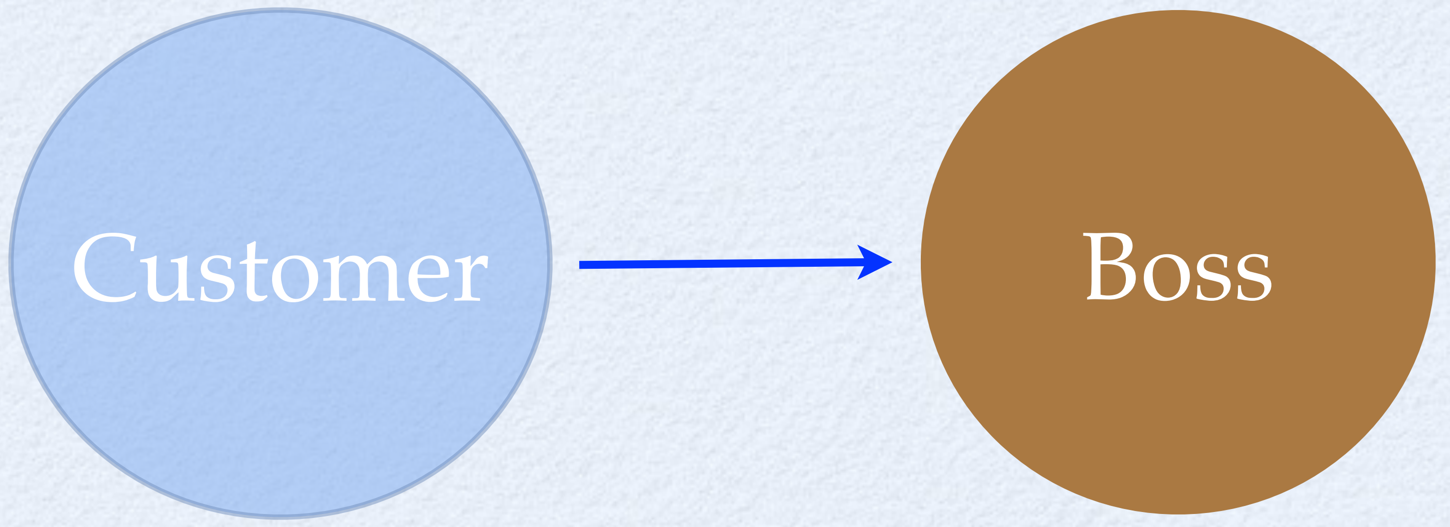


Example & Principles

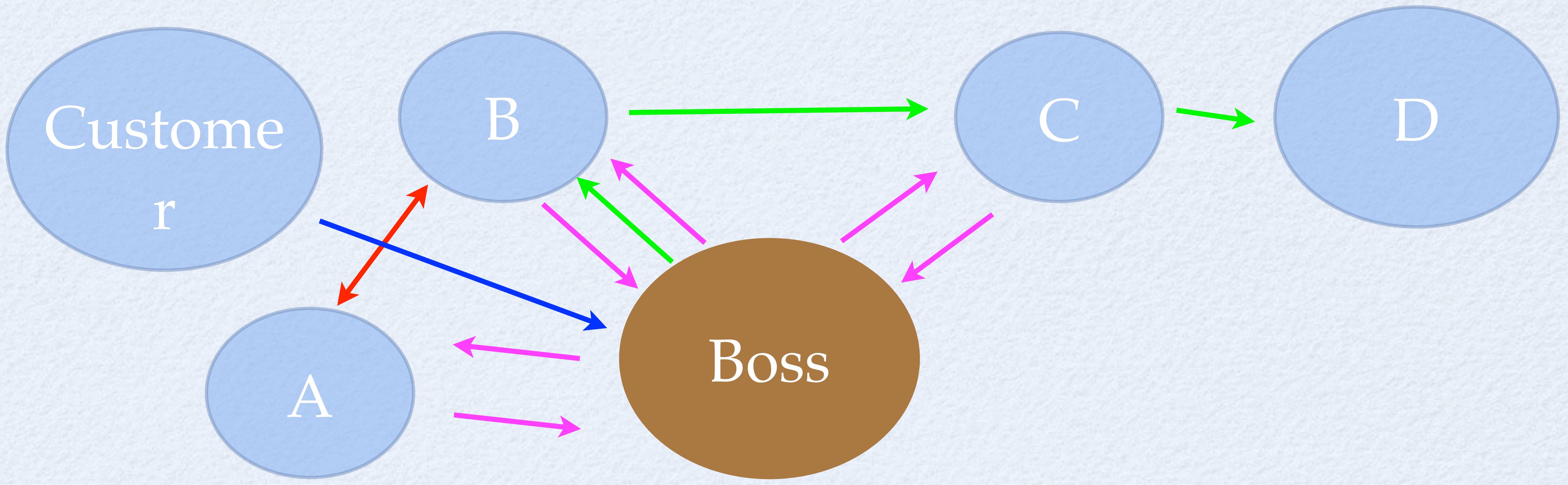


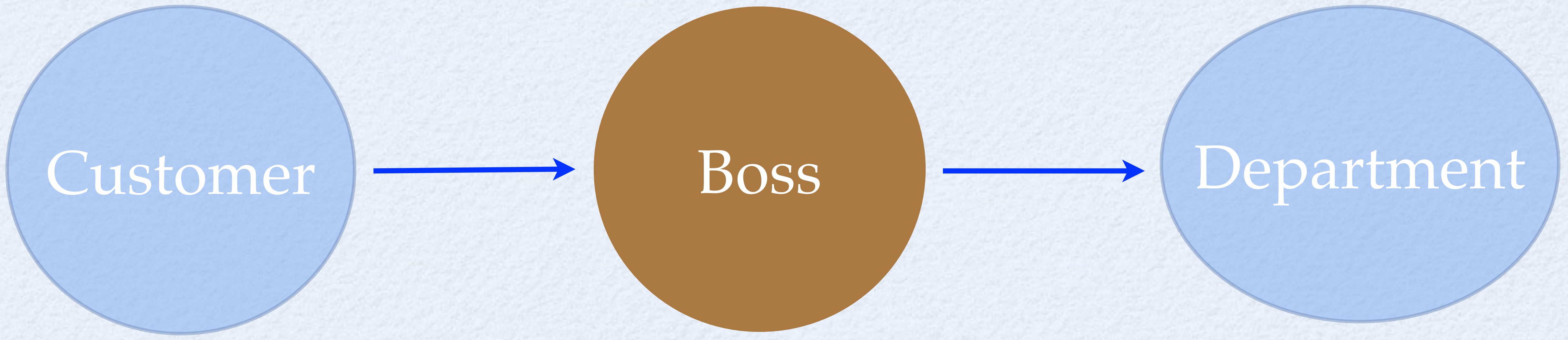
Example & Principles



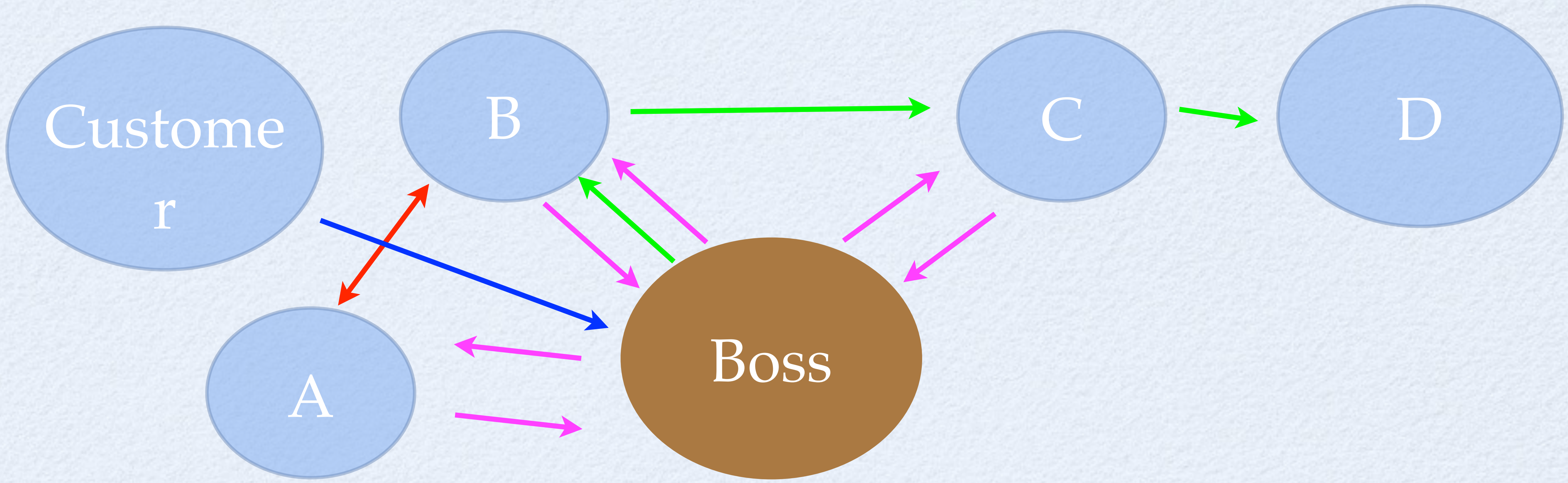


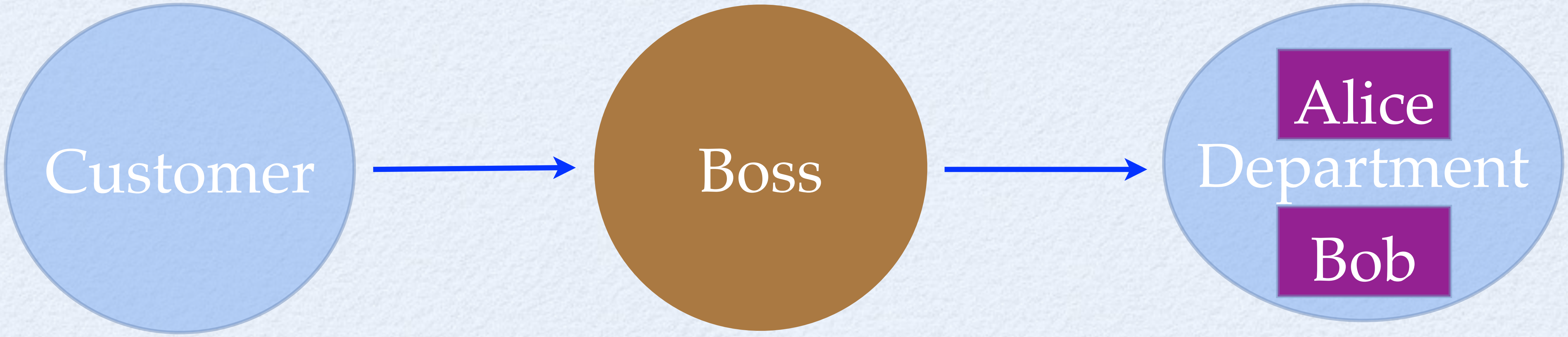
Example & Principles



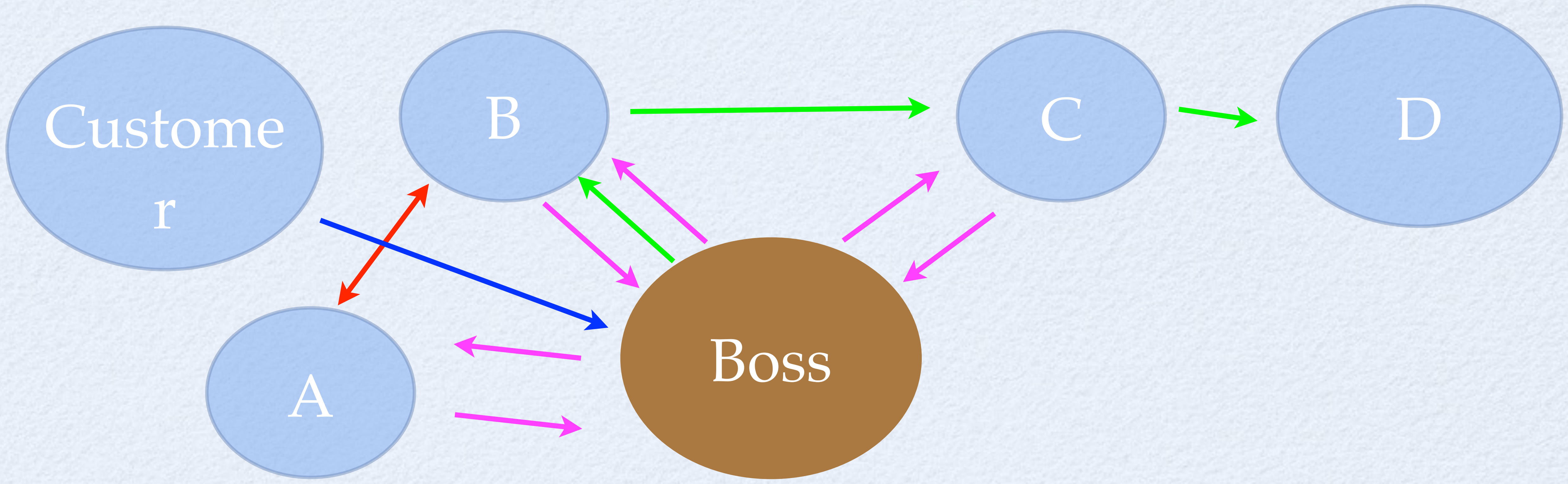


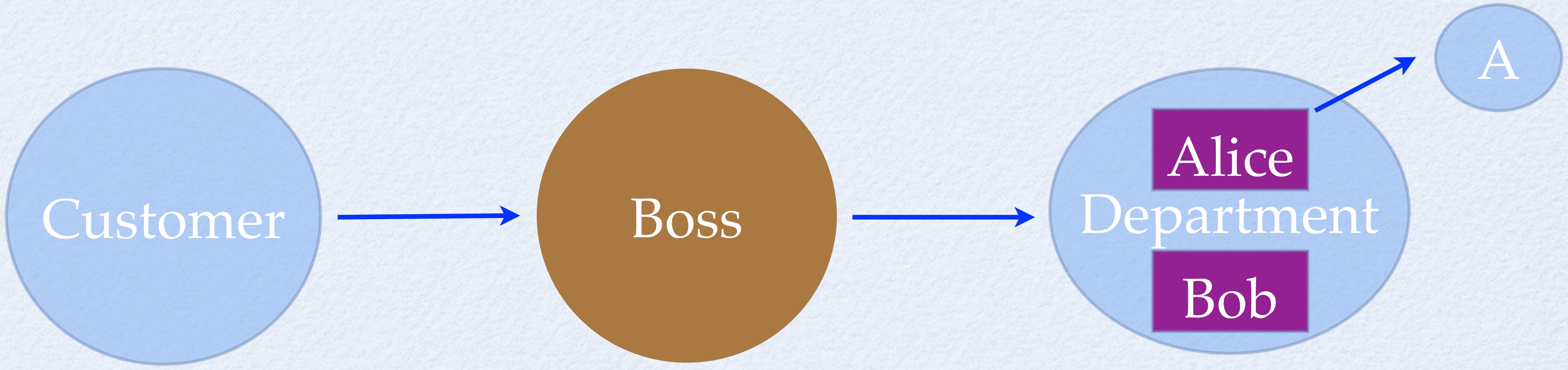
Example & Principles



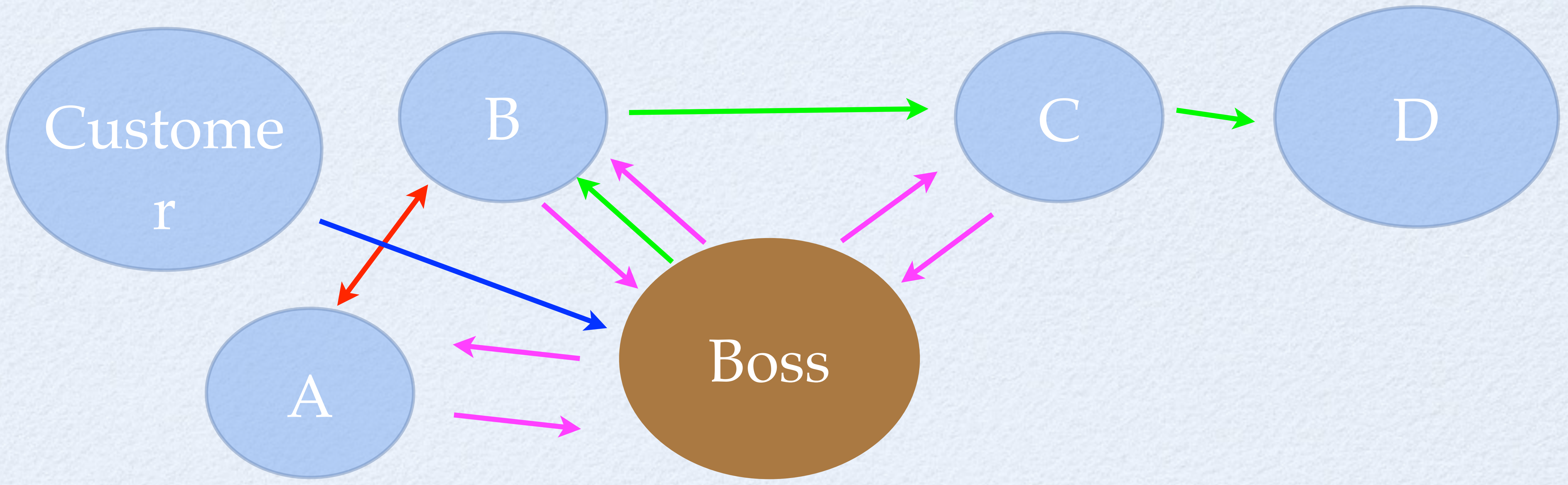


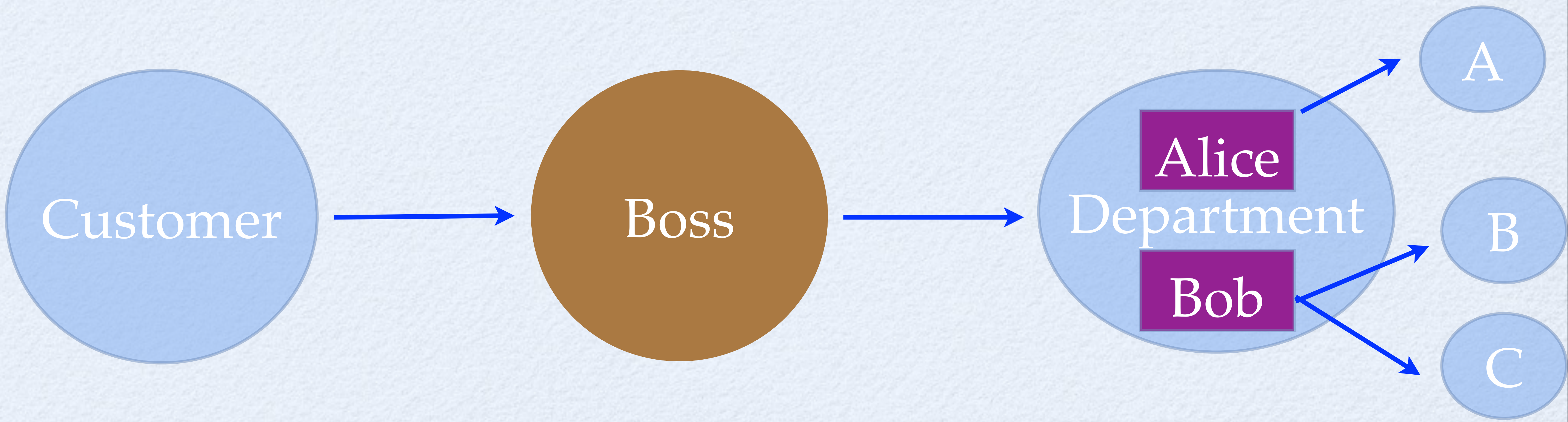
Example & Principles



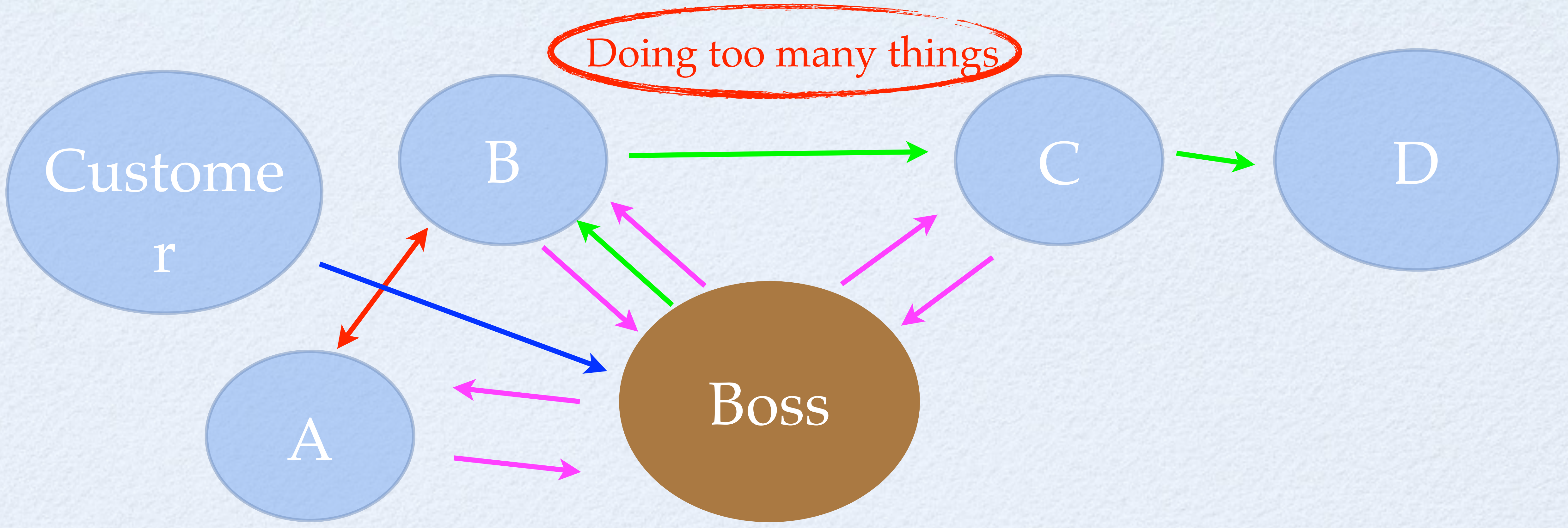


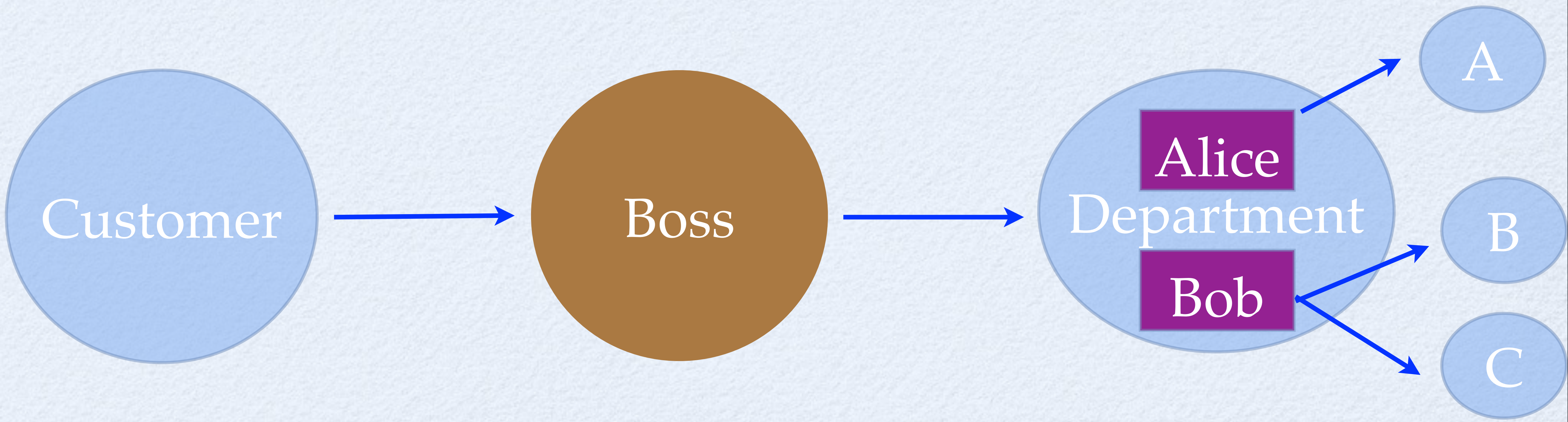
Example & Principles



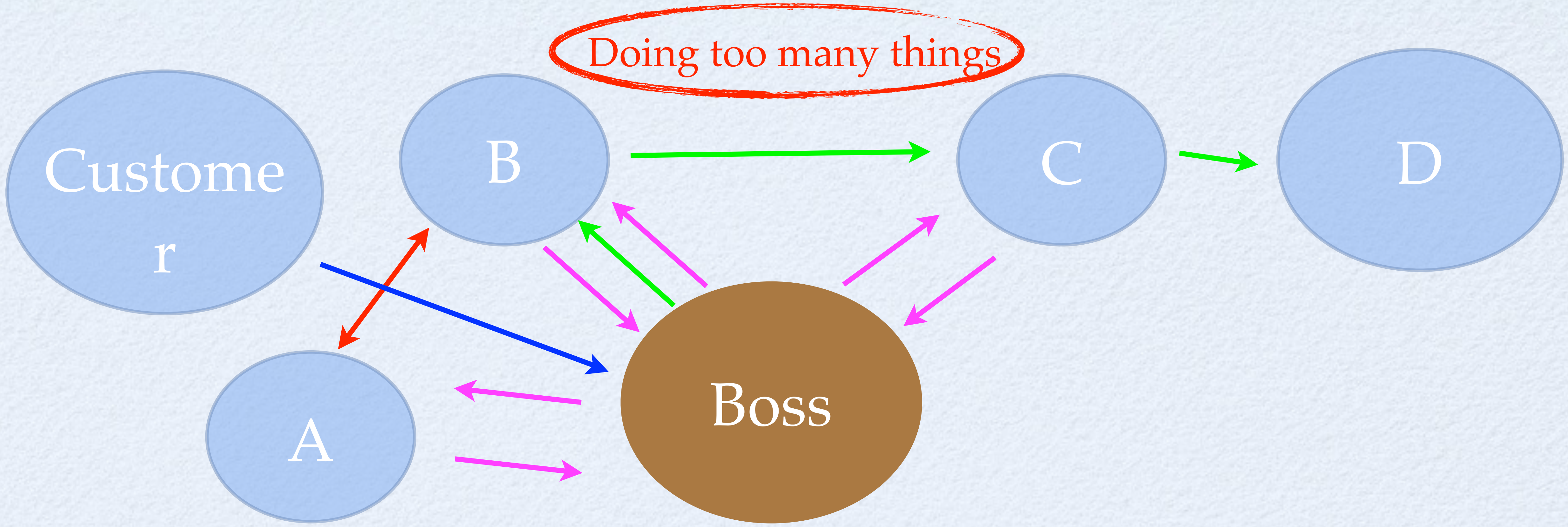


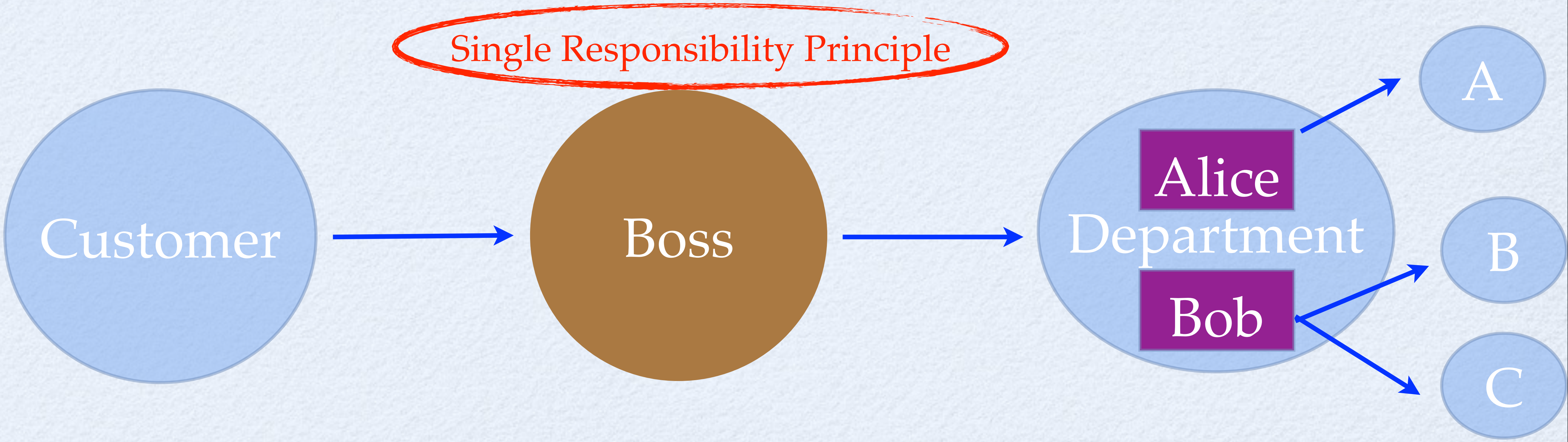
Example & Principles



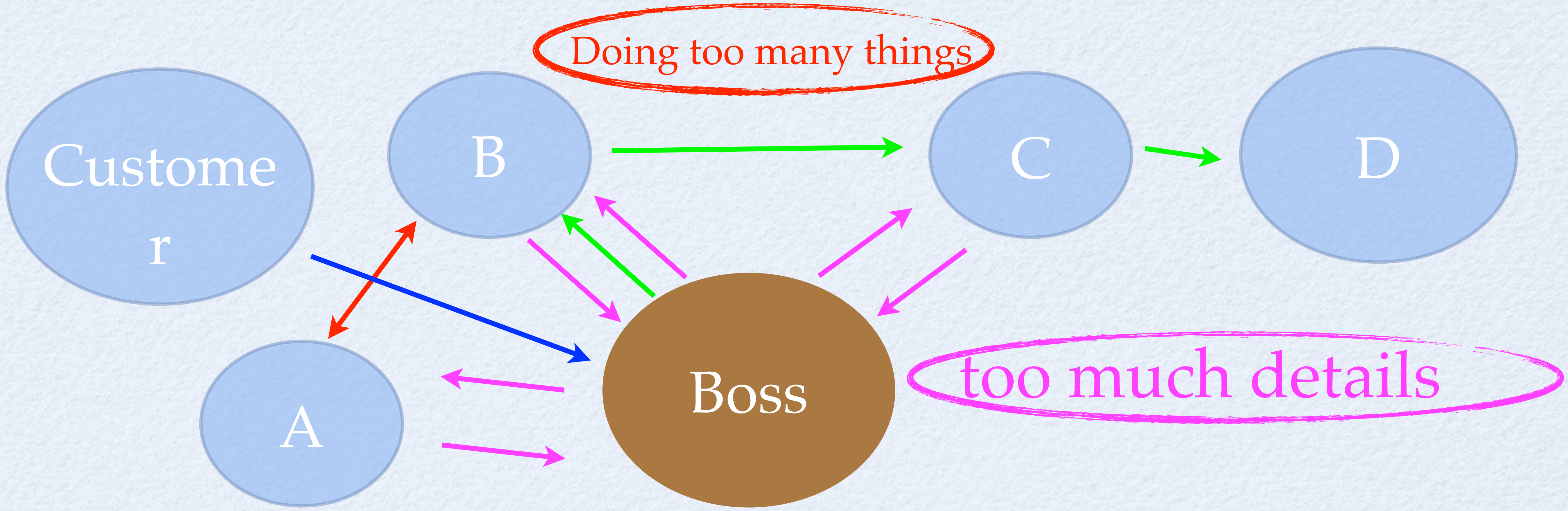


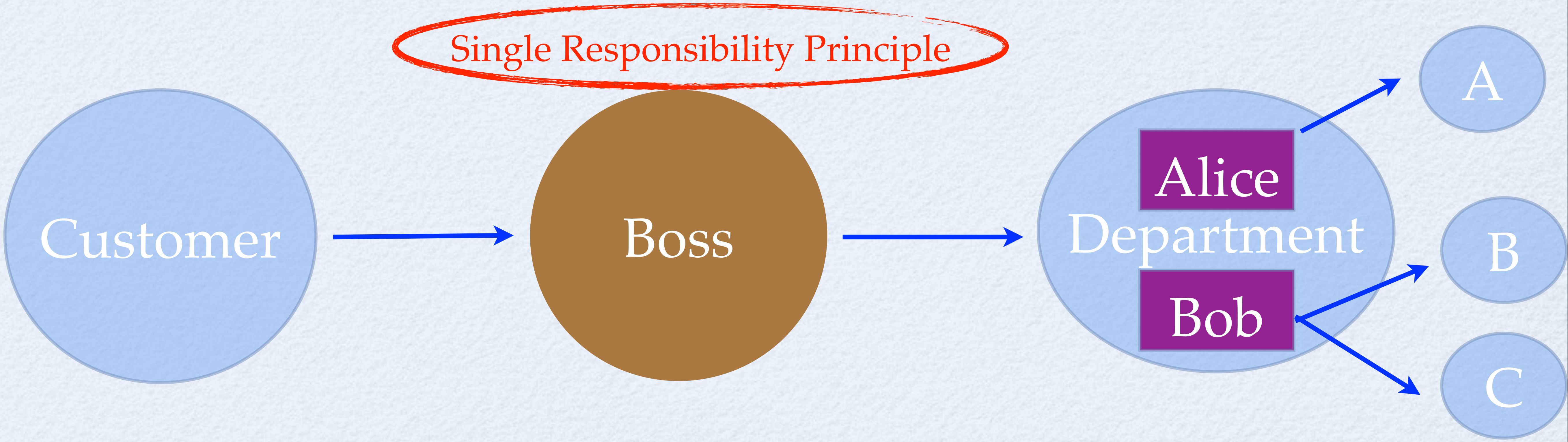
Example & Principles



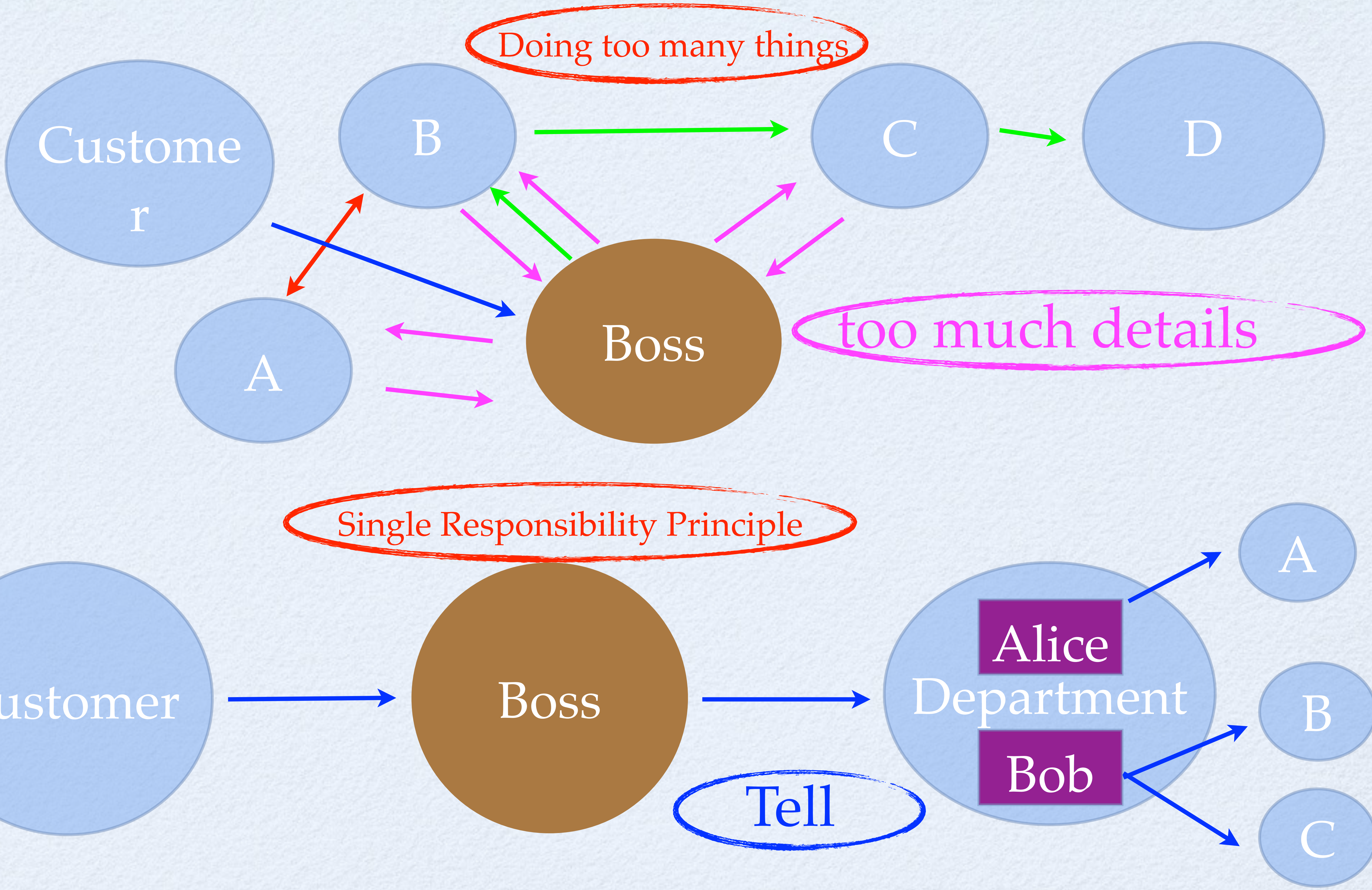


Example & Principles





Example & Principles



Review

Review

When do we to refactor code?

Review

When do we to refactor code?

What do we need to refactor?

Review

When do we to refactor code?

What do we need to refactor?

Test guides us go forward.

Review

When do we to refactor code?

What do we need to refactor?

Test guides us go forward.

We begin to refactor when we feel pain in Test.

Review

When do we to refactor code?

What do we need to refactor?

Test guides us go forward.

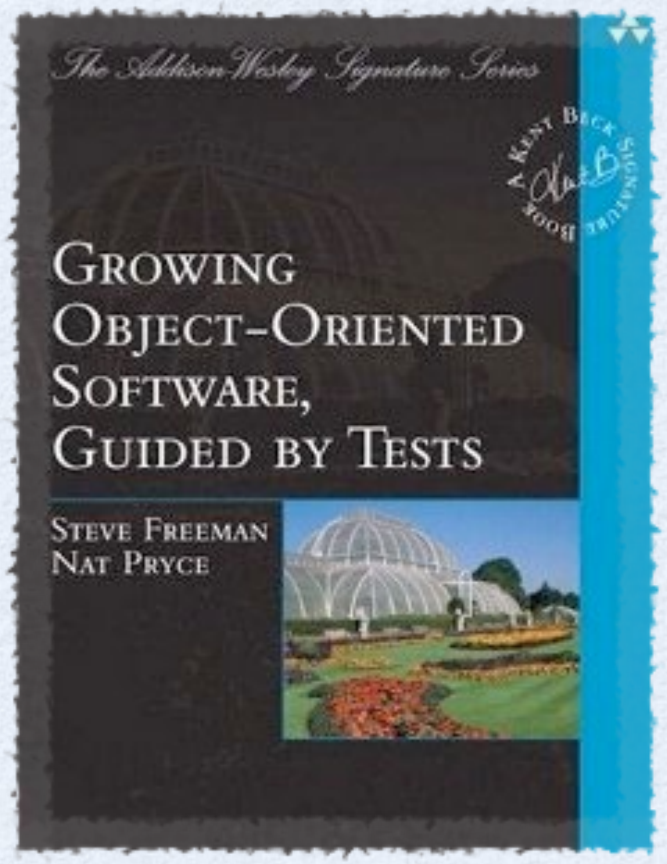
We begin to refactor when we feel pain in Test.

We reduce the dependencies found by Test.

Keep an eye on TEST.
Enjoy it!

Further Resources

GOOS


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce

POODR

Addison-Wesley Professional Ruby Series
PRACTICAL OBJECT-ORIENTED DESIGN IN RUBY
AN AGILE PRIMER
SANDI METZ
Foreword by Obie Fernandez

Tealeaf Academy
an online school for developers

Thanks!