

# About Architectures

Largely inspired by Robert C. Martin  
“Architecture: The Lost Years”

The screenshot shows a file explorer interface with two panes. The left pane displays the directory structure of a Ruby on Rails application named '\_updater'. The right pane shows the contents of the 'spec' directory, which is highlighted with a green background. The application structure includes:

- app
- assets
- controllers
- helpers
- mailers
- models
- views
- config
- db
- doc
- lib
- log
- public
- script
- spec
- tmp
- vendor
- .gitignore
- config.ru
- Gemfile
- Gemfile.lock
- Rakefile
- README.rdoc
- External Libraries

The 'spec' directory is expanded, showing files like \_\_init\_\_.py, app.py, database.py, extensions.py, models.py, and settings.py.

# Application intent not clear

The screenshot shows a file explorer interface with two panes. The left pane displays the directory structure of a Python project named 'libraryrest'. The right pane shows the contents of the 'libraryrest' directory, which is highlighted with a blue background. The project structure includes:

- libraryrest
- api
- \_\_init\_\_.py
- app.py
- database.py
- extensions.py
- models.py
- settings.py
- migrations
- requirements
- tests
- venv
- .gitignore
- .python-version
- .travis.yml
- LICENSE
- manage.py
- pytest.ini
- README.md
- requirements.txt
- Vagrantfile

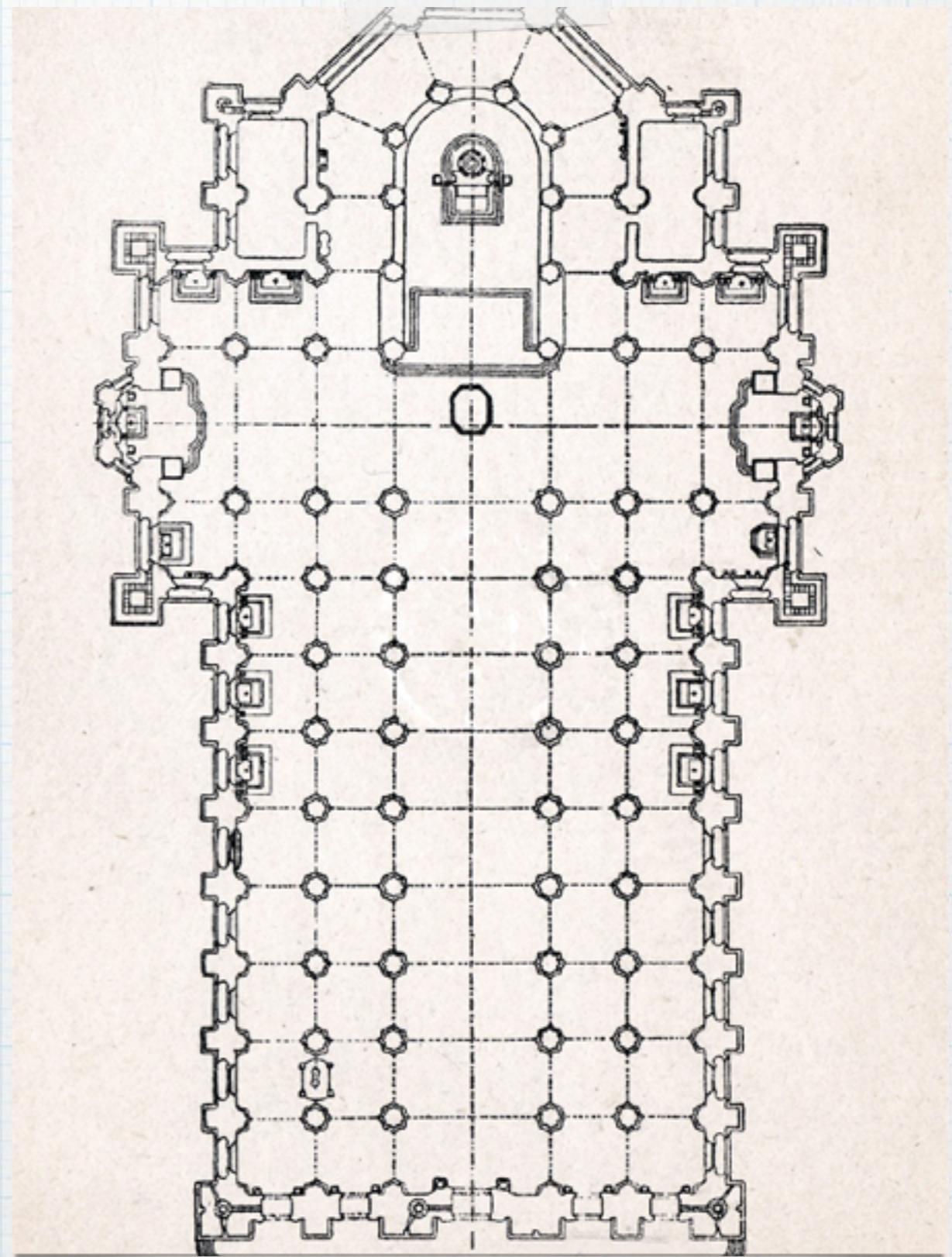
The WEB  
is a delivery mechanism

a common misconception:

“Web App”

# Architecture has to scream

the intent of the application



# Usecases

## USECASE

### Data

<user\_id>, ...

### Primary course

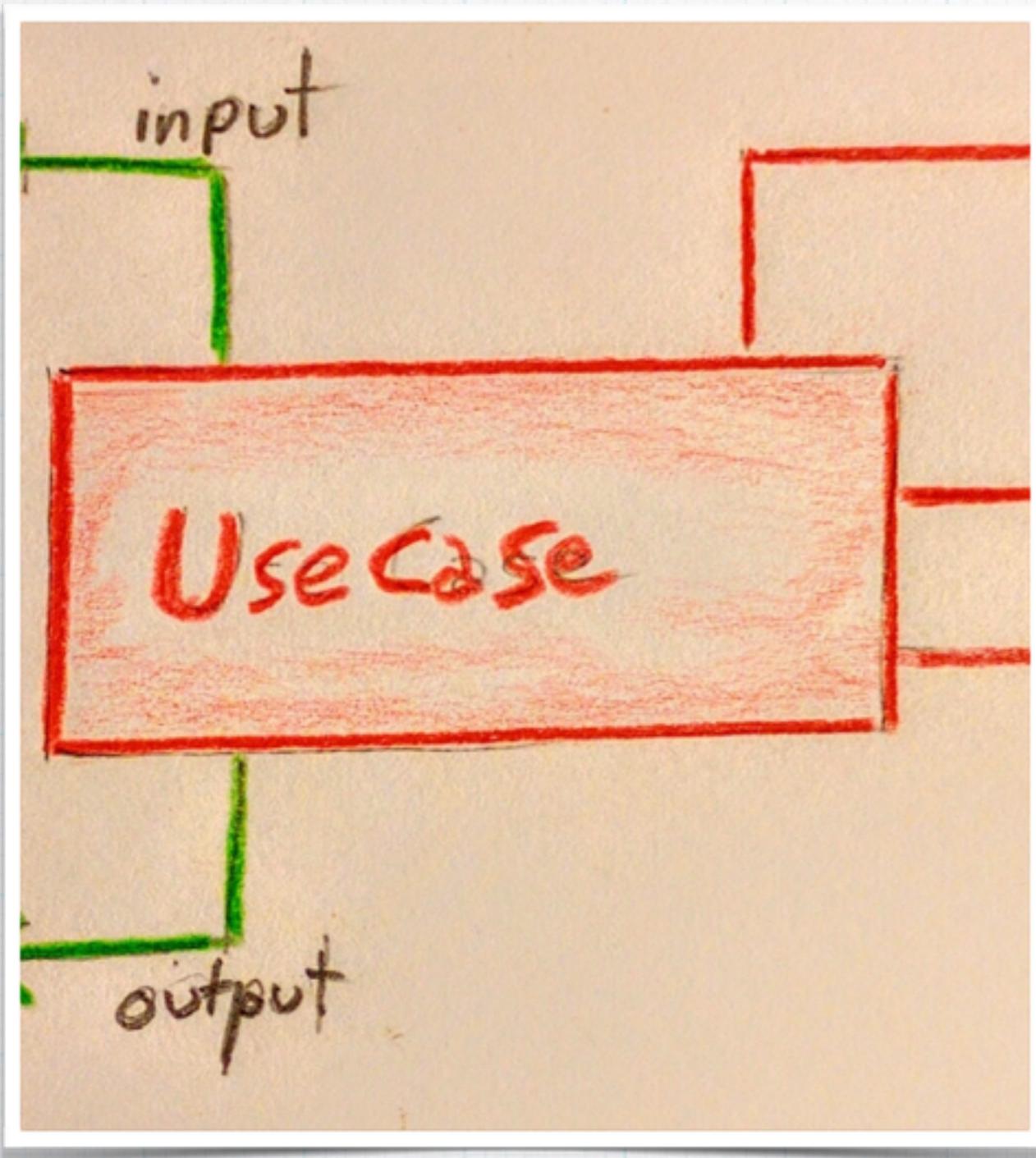
1. user issues create booking command
2. Request data validation
3. System confirm booking

### Exception course

1. system show error message ...

**Architecture of an application is driven by use cases**

Ivar Jacobson  
“Object Oriented Software Engineering”  
(1992)



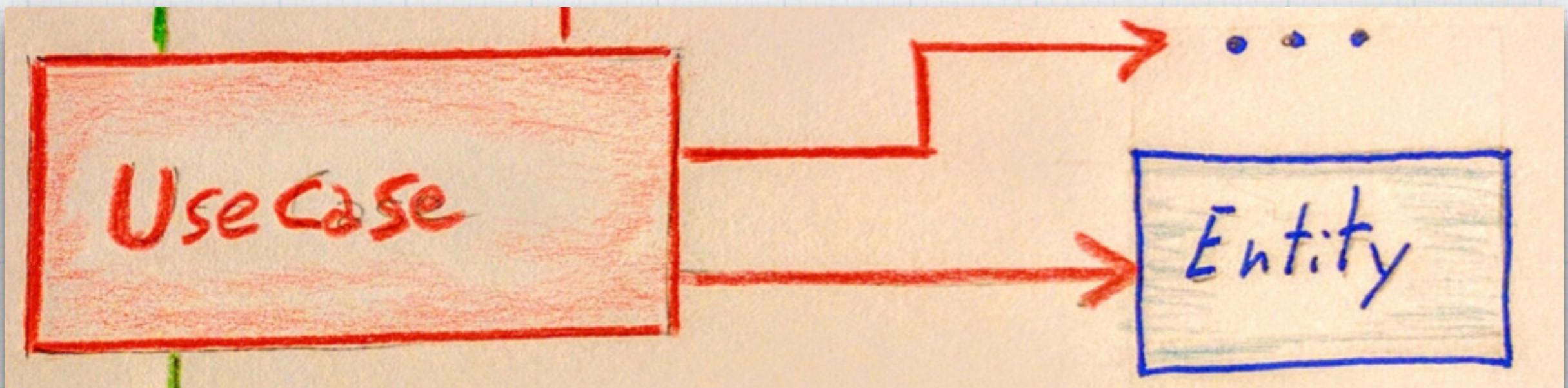
# Use case

Application specific  
business rules

e.g.:  
the booking process for  
your particular application

# Entity

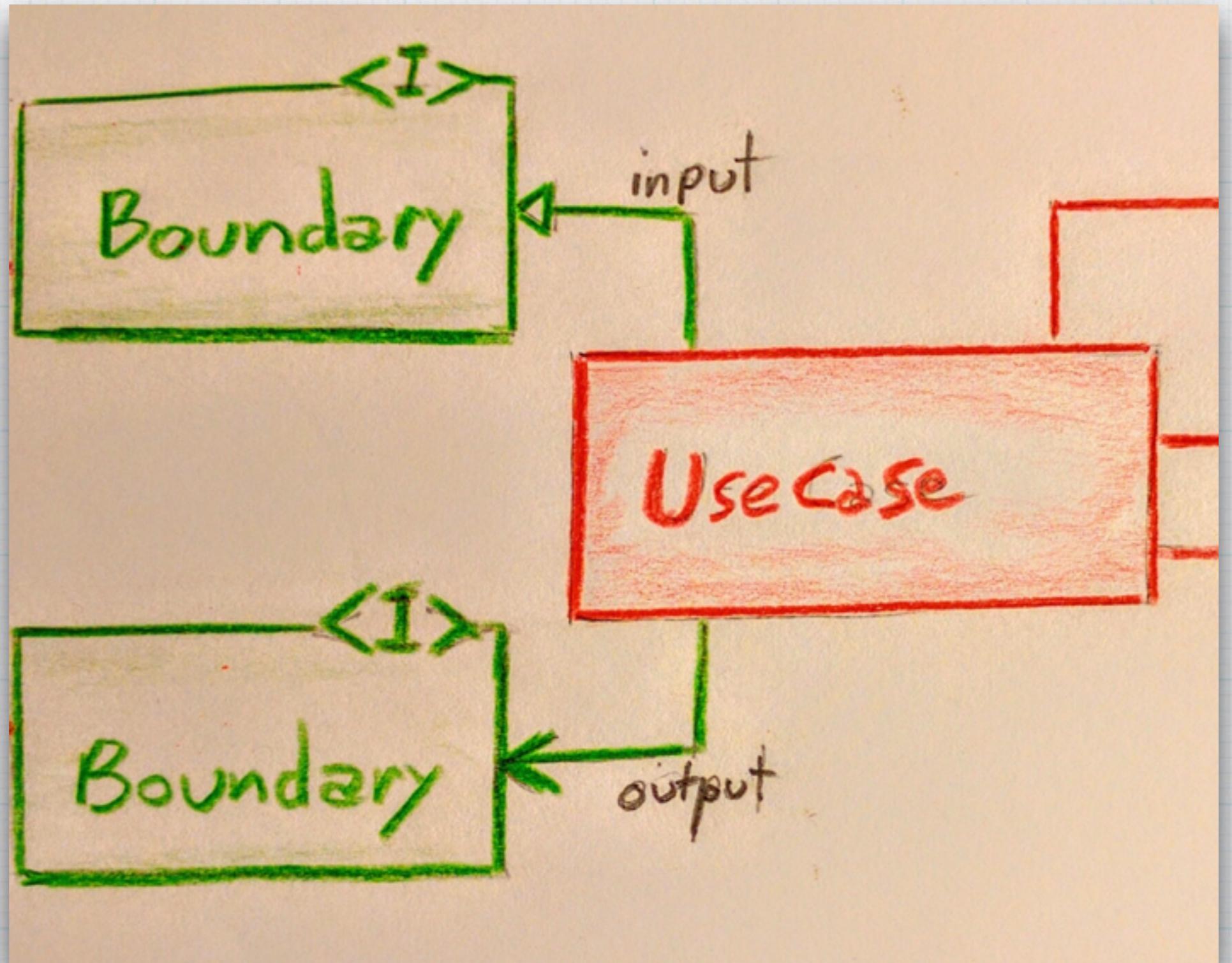
Application independent  
business rules



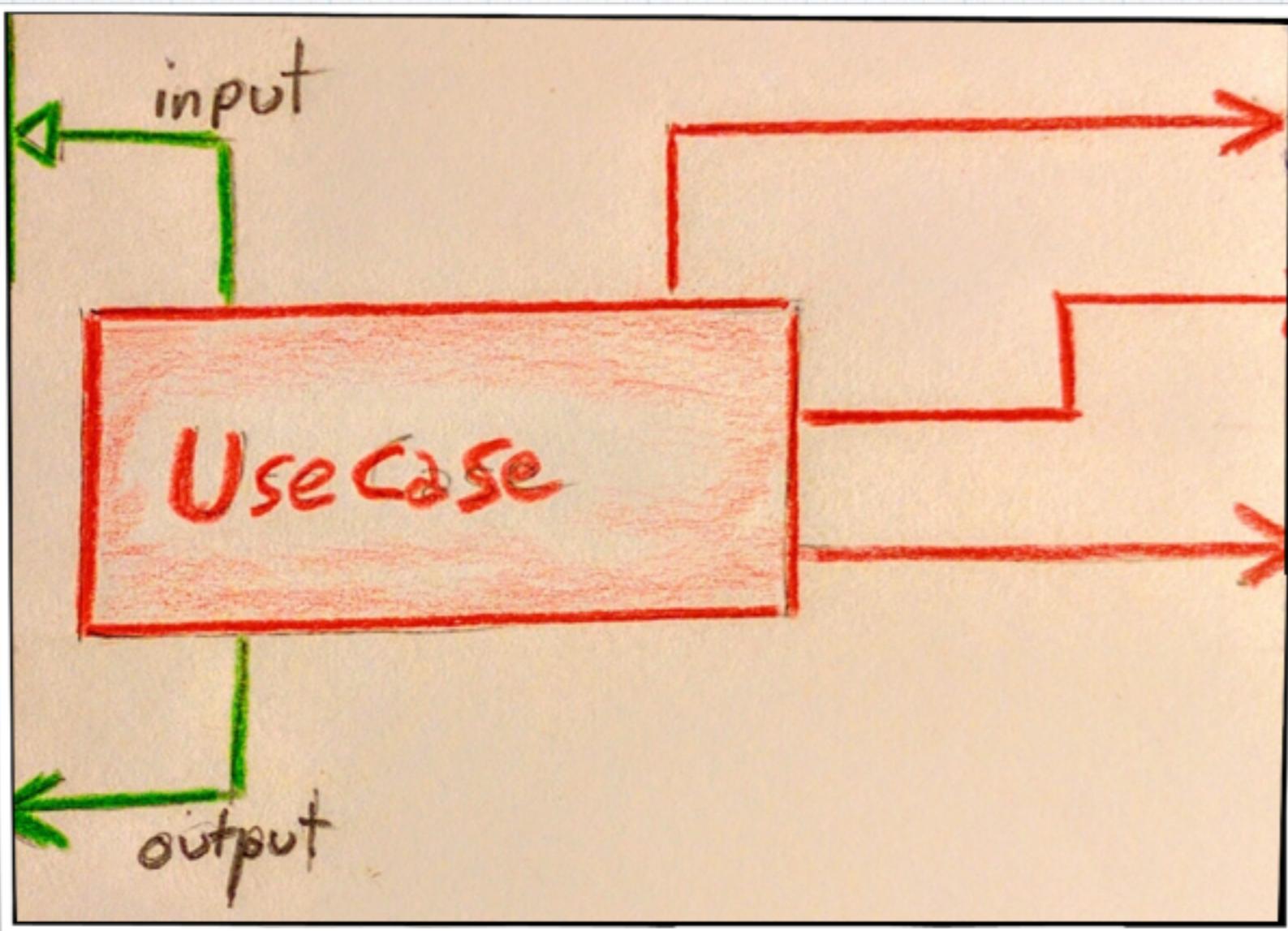
e.g.:  
the user entity composed  
by unique nickname etc...

# Boundaries

- input
- output

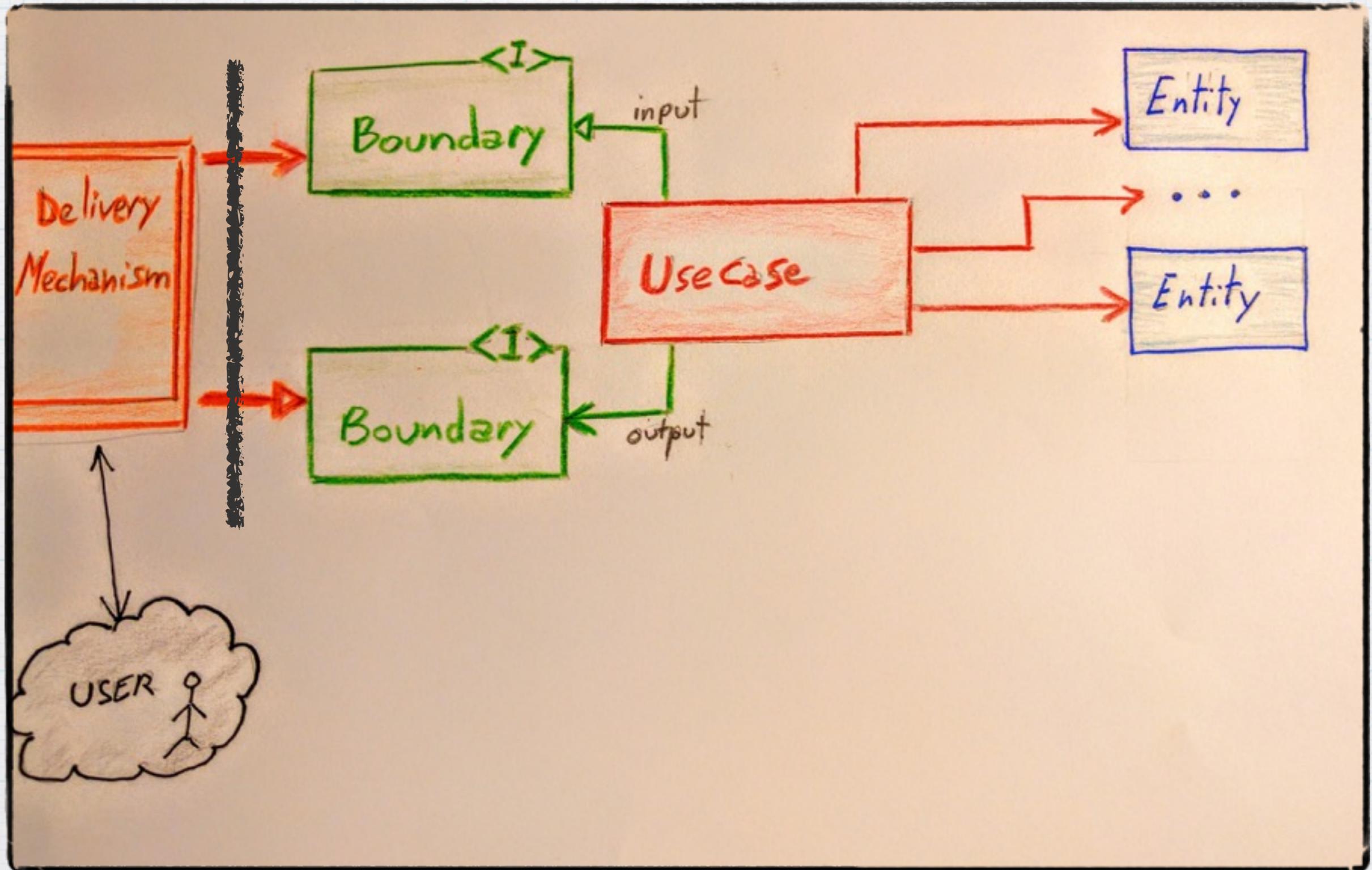


# Arrows

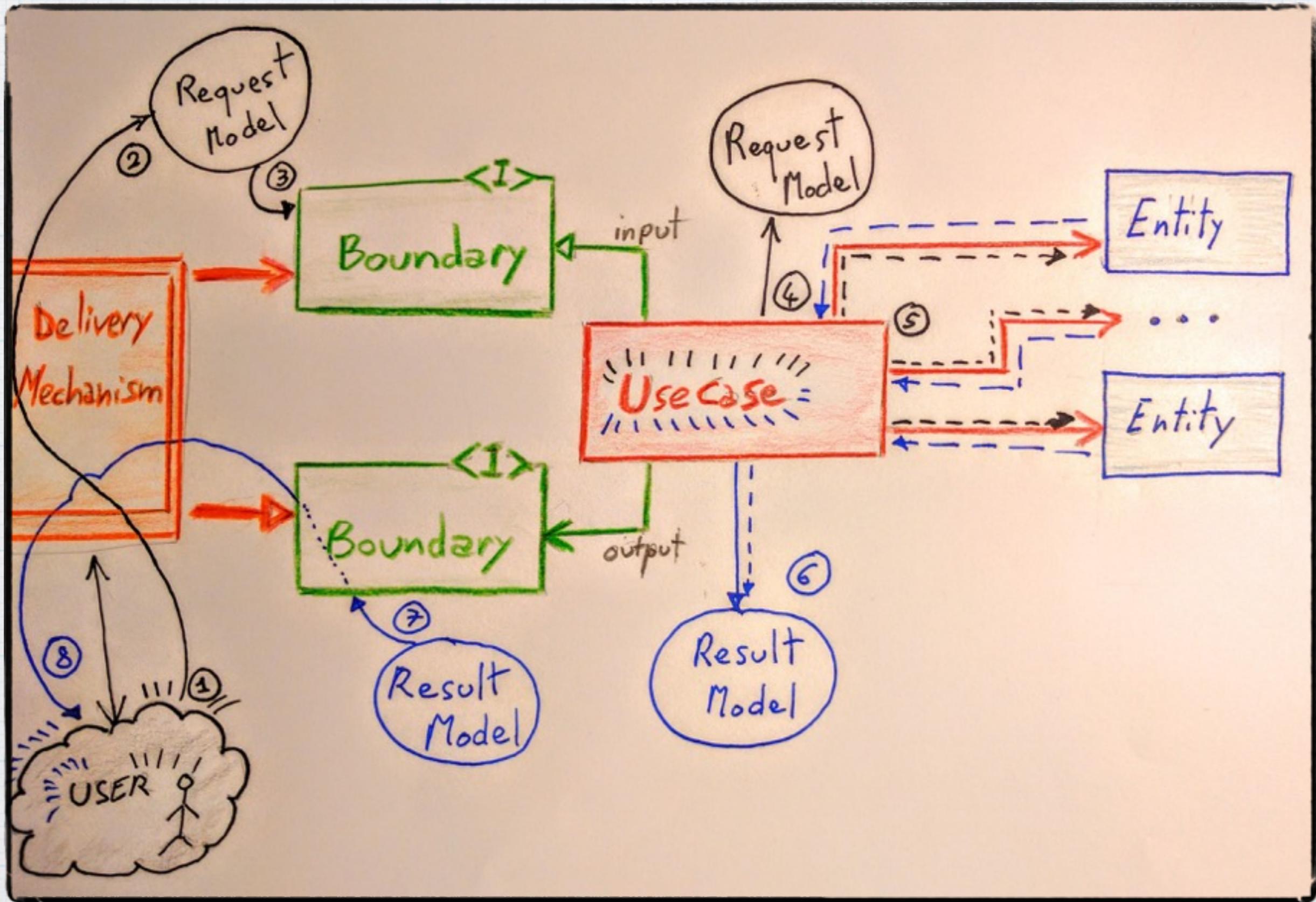


they point away from the usecase

# the big picture



# the flow



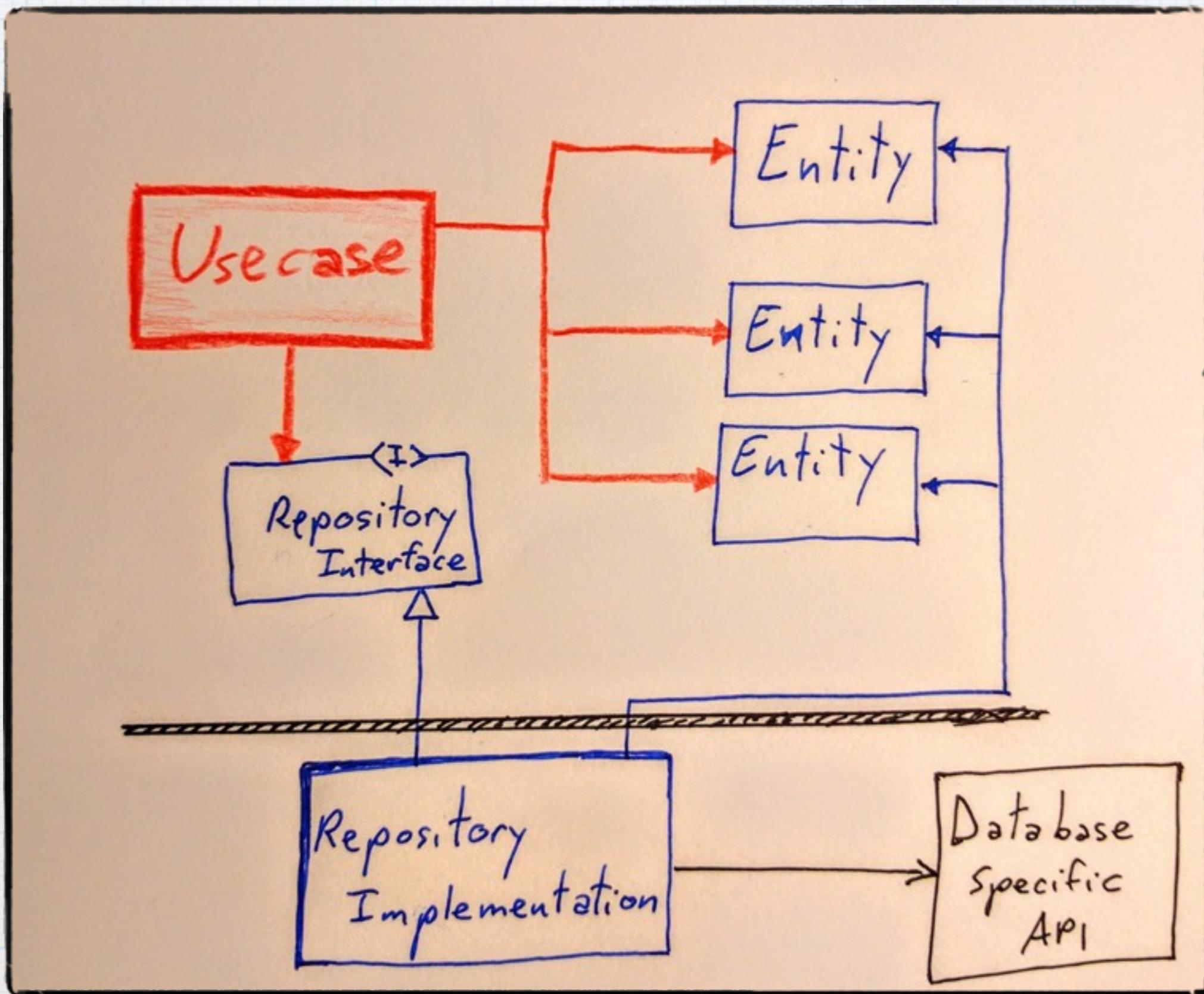
# Application reuse

- \* Application:
  - \* agnostic about delivery mechanism
  - \* can be used with different “communication channels”

# Testability

- \* Test in isolation without delivery mech.
- \* Use case business logic testable using Req and Res models

# the database



Design good  
architectures  
to  
maximize the number of  
deferred decisions

# Bibliography & Links

<https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html>

<http://www.taimila.com/blog/ddd-and-testing-strategy>

## Robert C. Martin videos

- Hakka Labs: Robert "Uncle Bob" Martin - Architecture: The Lost Years  
<https://www.youtube.com/watch?v=HhNIttd87xs>
- Ruby Midwest 2011: Architecture the Lost Years by Robert Martin  
<https://www.youtube.com/watch?v=WpkDN78P884>

Thanks!