Architecting for Tomorrow

a Pragmatic Perspective

Marcel Schutte - JFall, Ede - November 7th 2024

sogeti
Part of Capgemini

sogeti

Part of Capgemini

marcel.schutte@sogeti.com

linkedin.com/in/marceljschutte

@PragArchitect

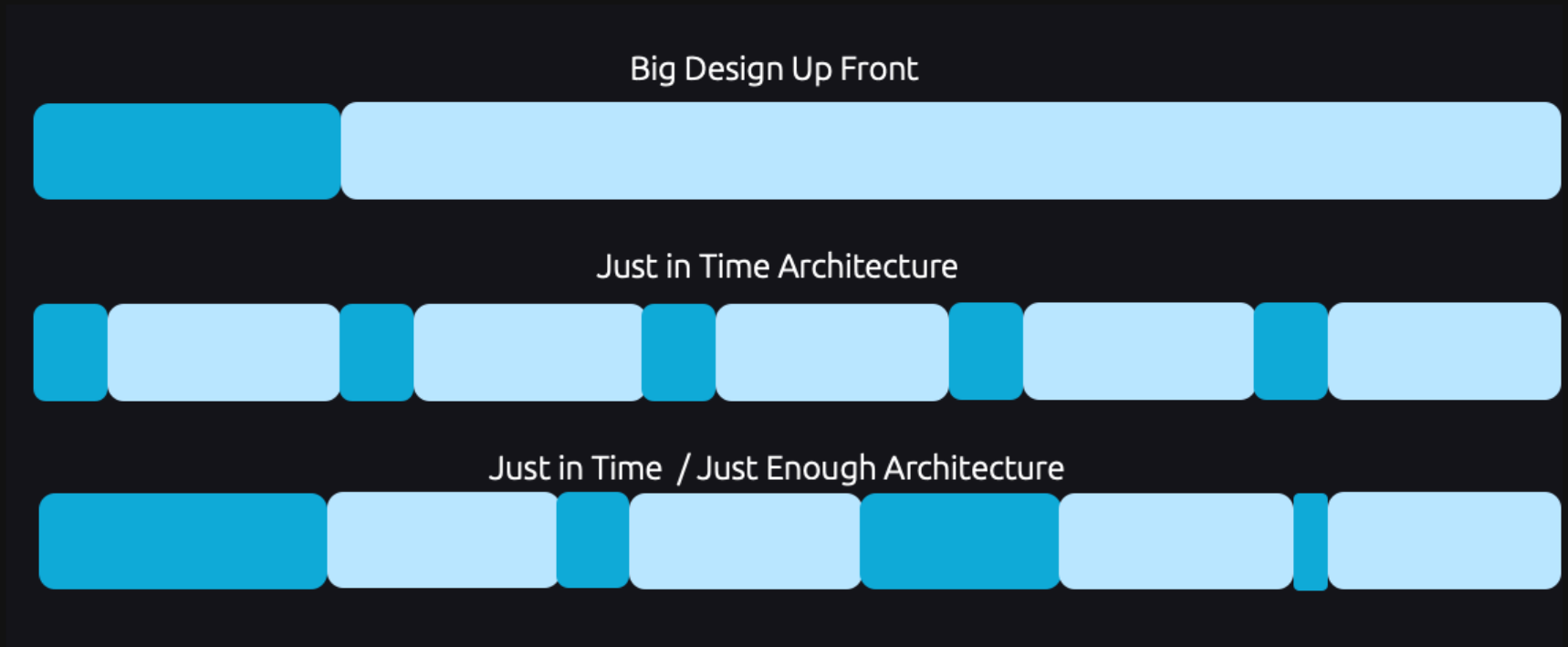
Architecture evolves

- Emergence of new Technologies
- Faster release cycles
- Agile
- Continuous Delivery & DevOps

Architecture evolves

- Emergence of new Technologies
- Faster release cycles
- Agile
- Continuous Delivery & DevOps

Agile Transformation



Good enough is the new perfect.

Anonymous

Evolutionary Design

Change is the constant

Therefore:

- Incremental Change
- Continuous Feedback
- Loose Coupling
- Resilient to Change

A complex system that works is invariably found to have evolved from a simple system that worked

John Gall

Introducing

the Pragmatic Architect

An architect who prioritises practical solutions that deliver tangible business value.

Pragmatic Architecture

Diagrams

- All Levels
- High Information Density
- Picture > 1000 words

C4 Model

Documentation

- Context & Knowledge
- Collaboration
- Agile Development

ADRs

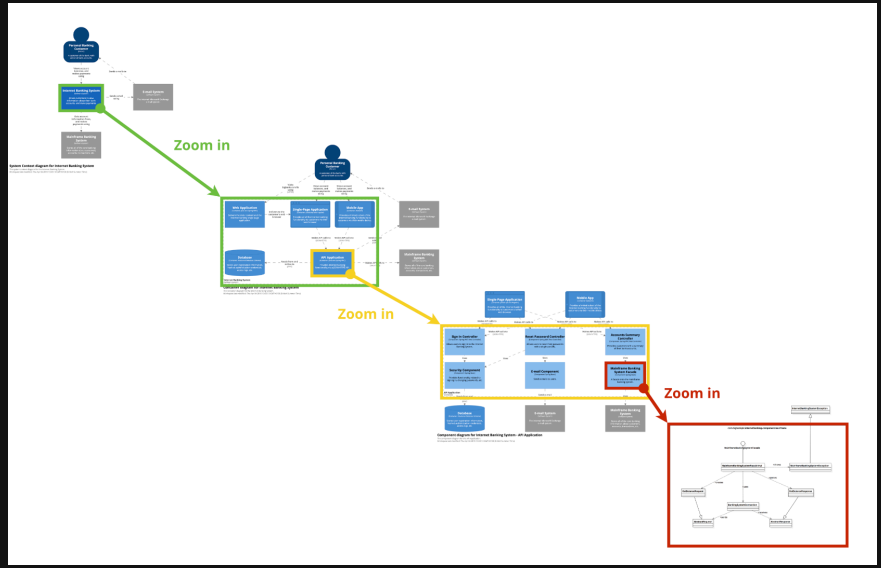
Fitness Functions

- Ensuring Adherence
- Guiding Evolution
- Automating Governance

ArchUnit

C4 Diagrams

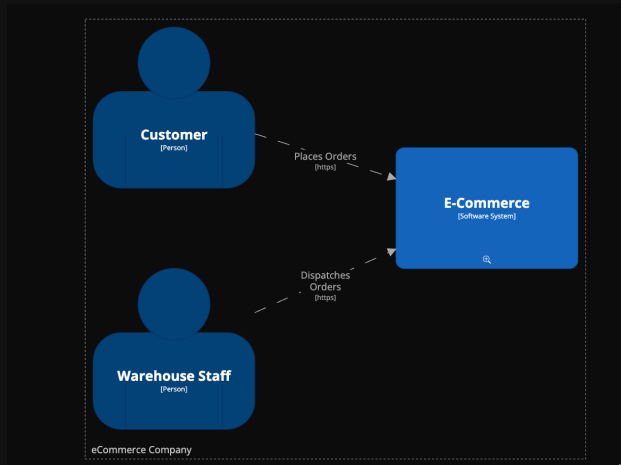
- Abstraction layers
- High information density
- Reusable



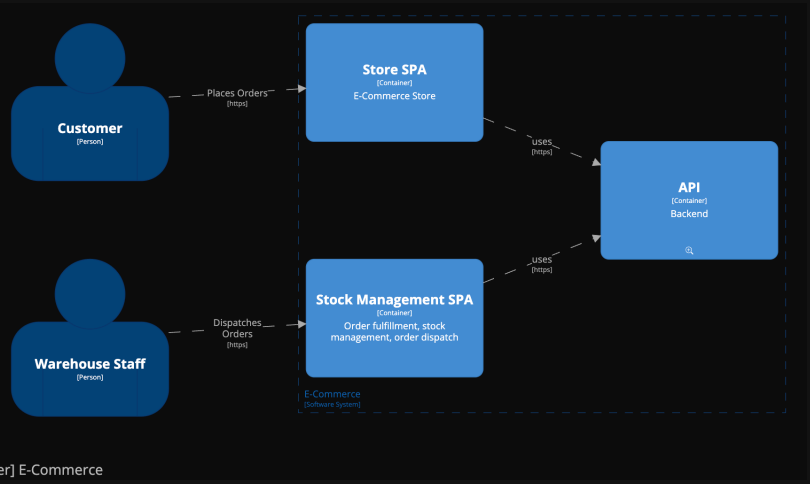
```
workspace {

    model {

        group "eCommerce Company" {

            customerPerson = person "Customer"
            warehousePerson = person "Warehouse Staff"

            ecommerceSystem = softwareSystem "E-Commerce" {
                storeContainer = container "Store SPA" "E-Commerce Store"
                stockContainer = container "Stock Management SPA" "Order fulfillment, stock management, order dispatch"

                apiContainer = container "API" "Backend"  {
                group "Web Layer" {
                    policyComp = component "Authorization Policy" "Authentication and authorization"
                    controllerComp = component "API Controller" "Requests, responses, routing and serialisation"
                    mediatrComp = component "MediatR" "Provides decoupling of requests and handlers"
                }

                }
            }
        }
```

```
# relationships between people and software systems
customerPerson -> storeContainer "Places Orders" "https"
warehousePerson -> stockContainer "Dispatches Orders" "https"

# relationships to/from containers
stockContainer -> apiContainer "uses" "https"
storeContainer -> apiContainer "uses" "https"


# relationships to/from components
storeContainer -> controllerComp "calls"
stockContainer -> controllerComp "calls"
controllerComp -> policyComp "authenticated and authorized by"
controllerComp -> mediatrComp "sends queries & commands to"
```

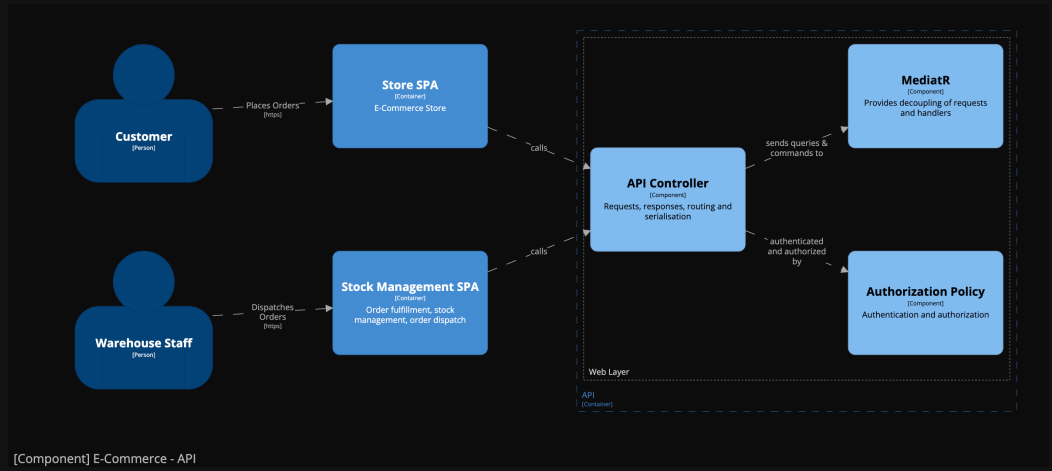
```
systemContext ecommerceSystem "SystemContext" {
    include *
    autoLayout lr
}

container ecommerceSystem "Container" {
    include *
    autoLayout lr
}

component apiContainer "Component" {
    include * customerPerson warehousePerson
    autoLayout lr
}
```



[System Context] E-Commerce



[Container] E-Commerce



[Component] E-Commerce - API

Architecture Decision Record

- Capture the WHY
- Preserve context

adr.github.io

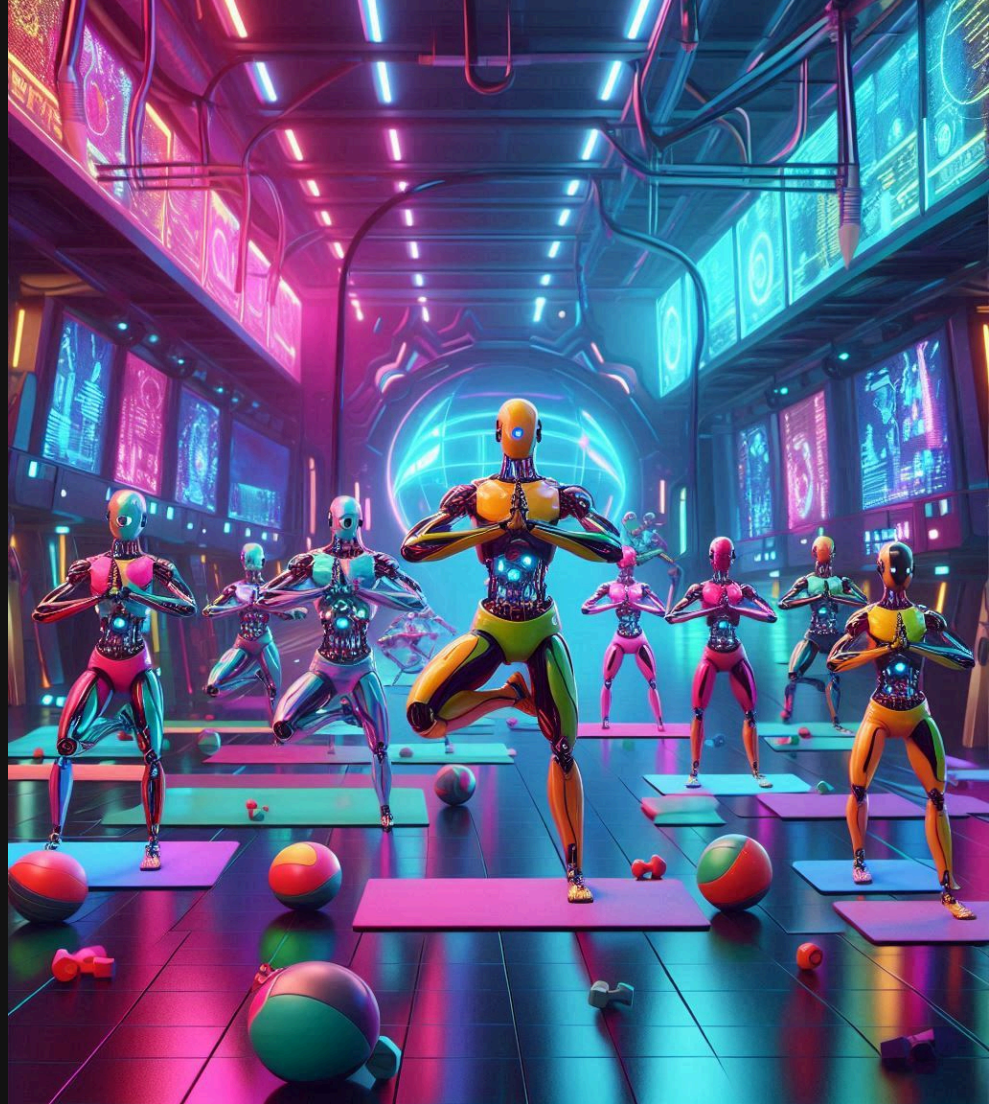
```
# These are optional elements. Feel free to remove any of them.
# status: "proposed | rejected | accepted | deprecated | … | superseded by ADR-0123"
# date: {YYYY-MM-DD when the decision was last updated}
# decision-makers: {list everyone involved in the decision}
# consulted: {list everyone whose opinions are sought (typically subject-matter experts); and with whom
# there is a two-way communication}
# informed: {list everyone who is kept up-to-date on progress; and with whom there is a one-way communication}

# {short title, representative of solved problem and found solution}

## Context and Problem Statement

<!-- optional element -->
## Decision Drivers

* {decision driver 1, e.g., a force, facing concern, …}
* {decision driver 2, e.g., a force, facing concern, …}
* … <!-- numbers of drivers can vary -->

## Considered Options

* {title of option 1}
* {title of option 2}
* {title of option 3}
```

adr.github.io

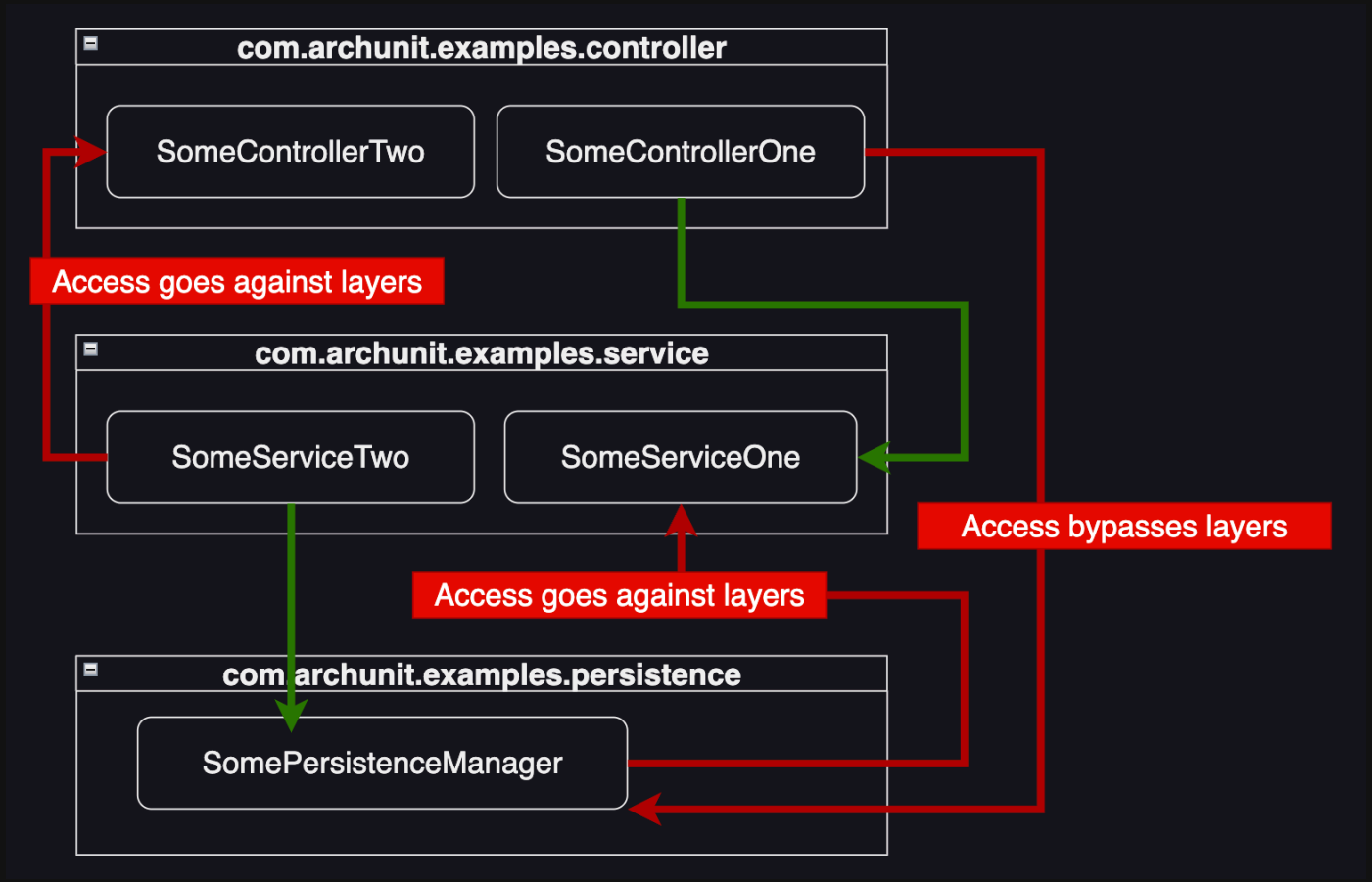
```
# {short title, representative of solved problem and found solution}

## Context and Problem Statement

## Considered Options

* {title of option 1}
* {title of option 2}
* {title of option 3}
* … <!-- numbers of options can vary -->

## Decision Outcome

Chosen option: "{title of option 1}", because {justification. e.g., only option,
which meets k.o. criterion decision driver | which resolves force {force} | … | comes out best (see below)}.

## Pros and Cons of the Options

### {title of option 1}

## More Information
```

ArchUnit

- Declarative Rules
- Automated Testing



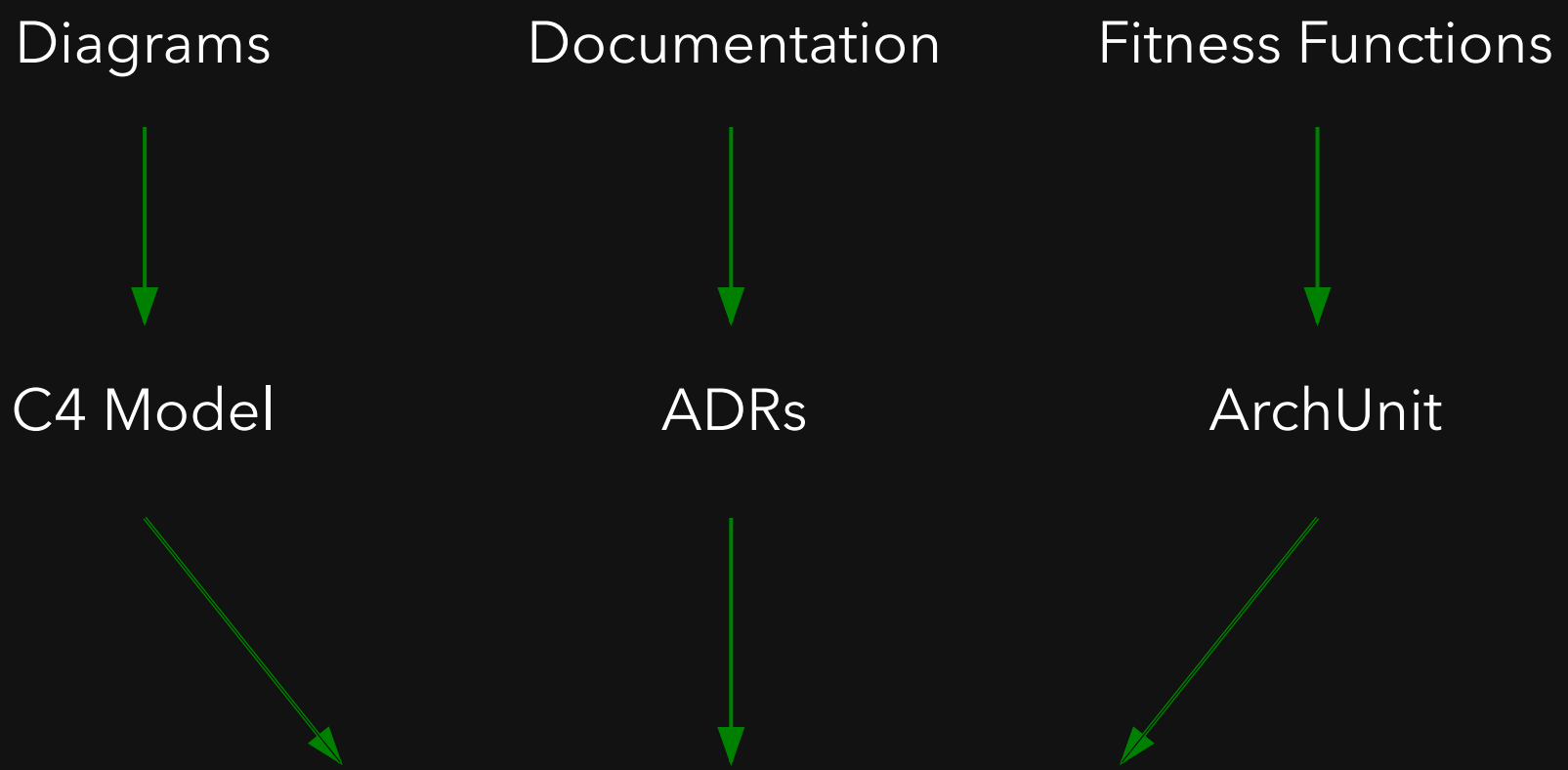
ArchUnit


com.archunit.examples.controller
SomeControllerTwo
SomeControllerOne
Access goes against layers
com.archunit.examples.service
SomeServiceTwo
SomeServiceOne
Access bypasses layers
Access goes against layers
com.archunit.examples.persistence
SomePersistenceManager

ArchUnit

```
JavaClasses jc = new ClassFileImporter()
  .importPackages("com.archunit.examples");

LayeredArchitecture arch = layeredArchitecture()
   // Define layers
  .layer("Presentation").definedBy("..controller..")
  .layer("Service").definedBy("..service..")
  .layer("Persistence").definedBy("..persistence..")
  // Add constraints
  .whereLayer("Presentation").mayNotBeAccessedByAnyLayer()
  .whereLayer("Service").mayOnlyBeAccessedByLayers("Presentation")
  .whereLayer("Persistence").mayOnlyBeAccessedByLayers("Service");
arch.check(jc);
```


RECAP


Diagrams
Documentation
Fitness Functions
C4 Model
ADRs
ArchUnit
Pragmatic Architecture

Thank you for your attention!


marcel.schutte@sogeti.com


linkedin.com/in/marceljschutte


@PragArchitect

Slides can be found here:

https://github.com/marceljschutte/slidedecks/

END