AI-Infused Software Architecture

building the Future

Marcel Schutte - aiGrunn, Groningen - November 29th 2024

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

AI will not replace architects, but architects who use AI will replace those who don’t.

Software Architecture

Diagrams

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

C4 Model

Documentation

- Context & Knowledge
- Collaboration
- Agile Development

ADRs

Fitness Functions

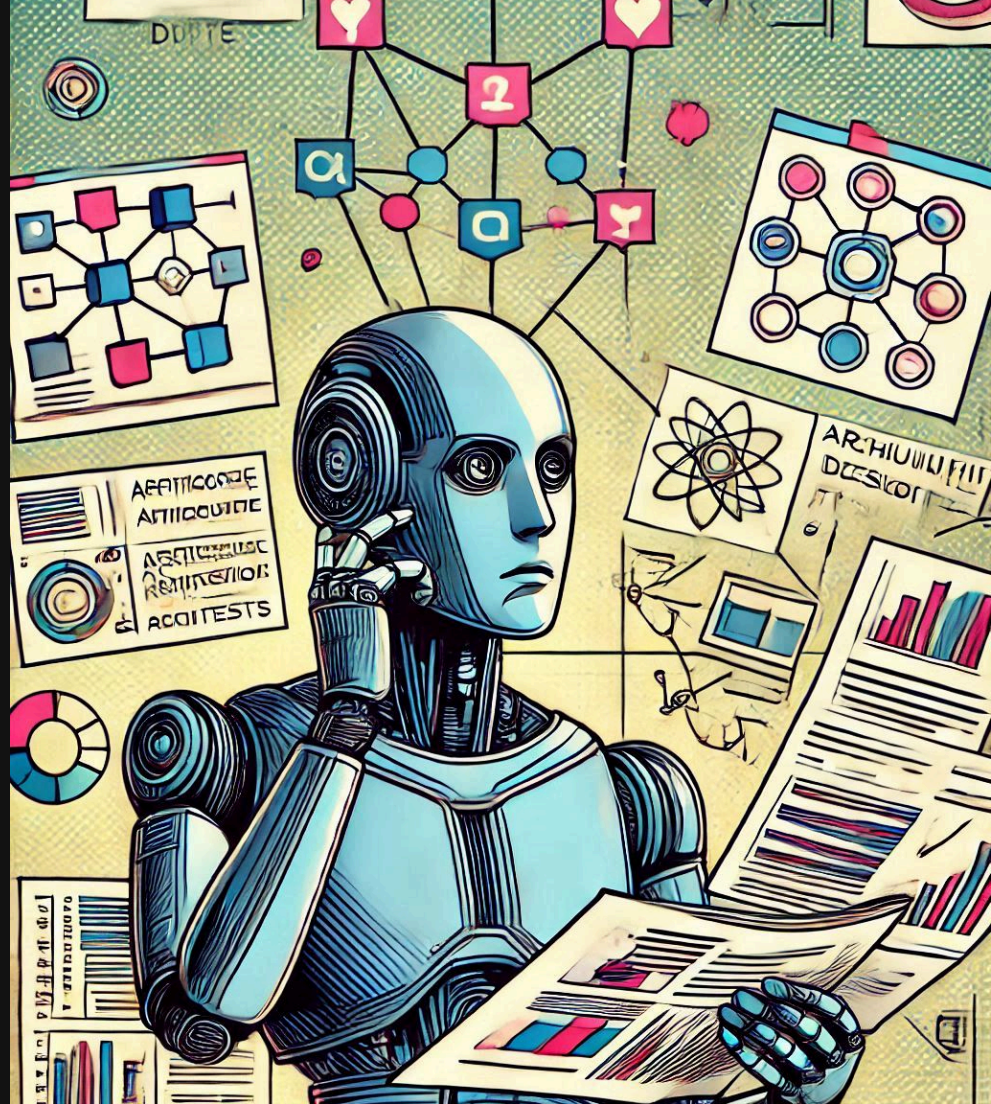
- Ensuring Adherence
- Guiding Evolution
- Automating Governance

ArchUnit

Perfect for GenAI

- C4 => DSL
- ADR's => Clear templates
- ArchUnit => Code

All text based



Pizza restaurant "Il Sogno Italiano"

a Case study

Pizzeria ”Il Sogno Italiano" is a family-owned business that has been making fresh pizzas with passion for over 20 years. The restaurant is beloved in the area for its authentic recipes, fresh ingredients, and cozy atmosphere.

The pizzeria wants to optimize its online ordering process. The current website is outdated and cumbersome, resulting in customer frustration and a loss of potential orders.

The pizzeria wants a user-friendly ordering application that simplifies the ordering process and increases online orders.

Prompt 1 - The setup

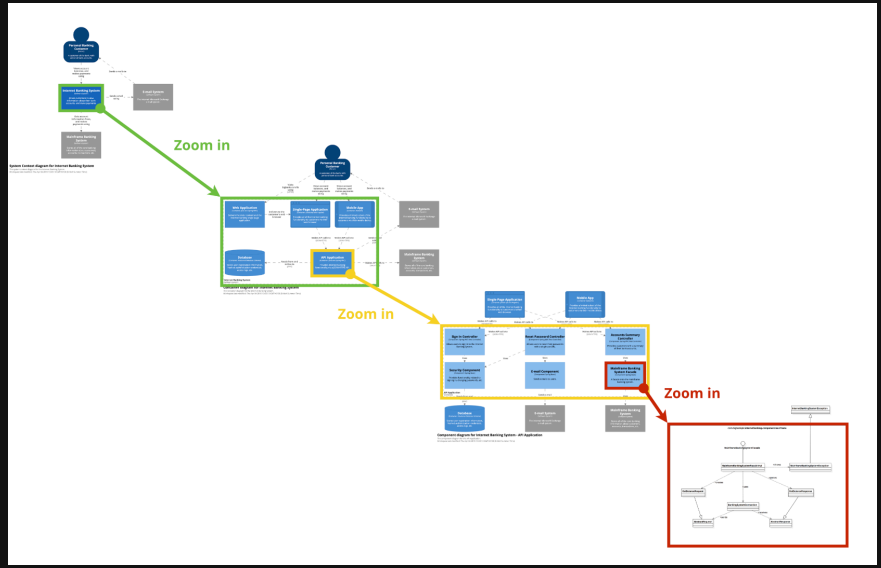
```
I am a software architect working on developing a new software architecture.

I need help creating a high-level structure that includes key components, their responsibilities, and interactions.

Can you guide me through this step by step or provide suggestions tailored to modern best practices?
```

Prompt 2 - Project initialization

```
"I am designing an online pizza ordering tool for a local pizza restaurant.
Customers can browse the menu and place orders. Bakers need to track baking tasks, and delivery drivers
should receive notifications when pizzas are ready and see delivery addresses. The backend uses Java
with a PostgreSQL database, but the frontend technology is undecided. Can you help me:

1. Design a scalable architecture for this system, ensuring clear responsibilities for each component?
2. Propose frontend technologies that integrate well with the backend?
3. Suggest database schema ideas to support customers, orders, and delivery logistics?"
```

C4 Diagrams

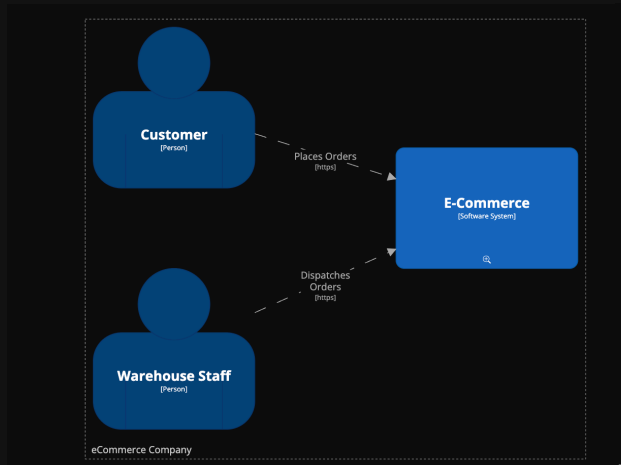
- Abstraction layers
- High information density
- Reusable



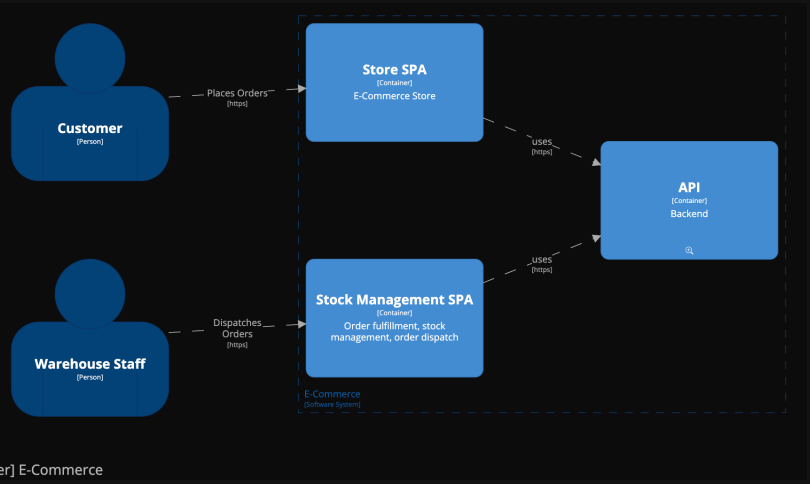
```
workspace {

    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"
```

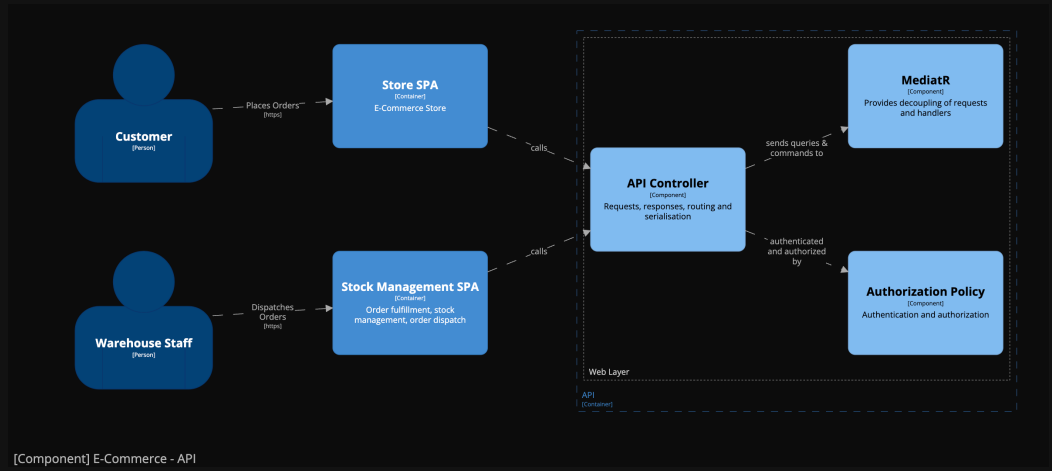
```
systemContext ecommerceSystem "SystemContext" {
    include *
    autoLayout lr
}

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

Prompt 3 - C4 modelling

```
I want to create a C4 model using Structurizr that includes:

1. A Context diagram showing interactions with customers, bakers, delivery drivers, and external payment systems.
2. A Container diagram illustrating frontend, backend, and database components.
3. A Component diagram detailing backend services (e.g., Order Service, Baking Service).

Can you help generate Structurizr-compatible code and ensure the diagrams reflect good architecture practices,
including clarity and scalability?"
```

Architecture Decision Record

- Capture the WHY
- Preserve context

adr.github.io

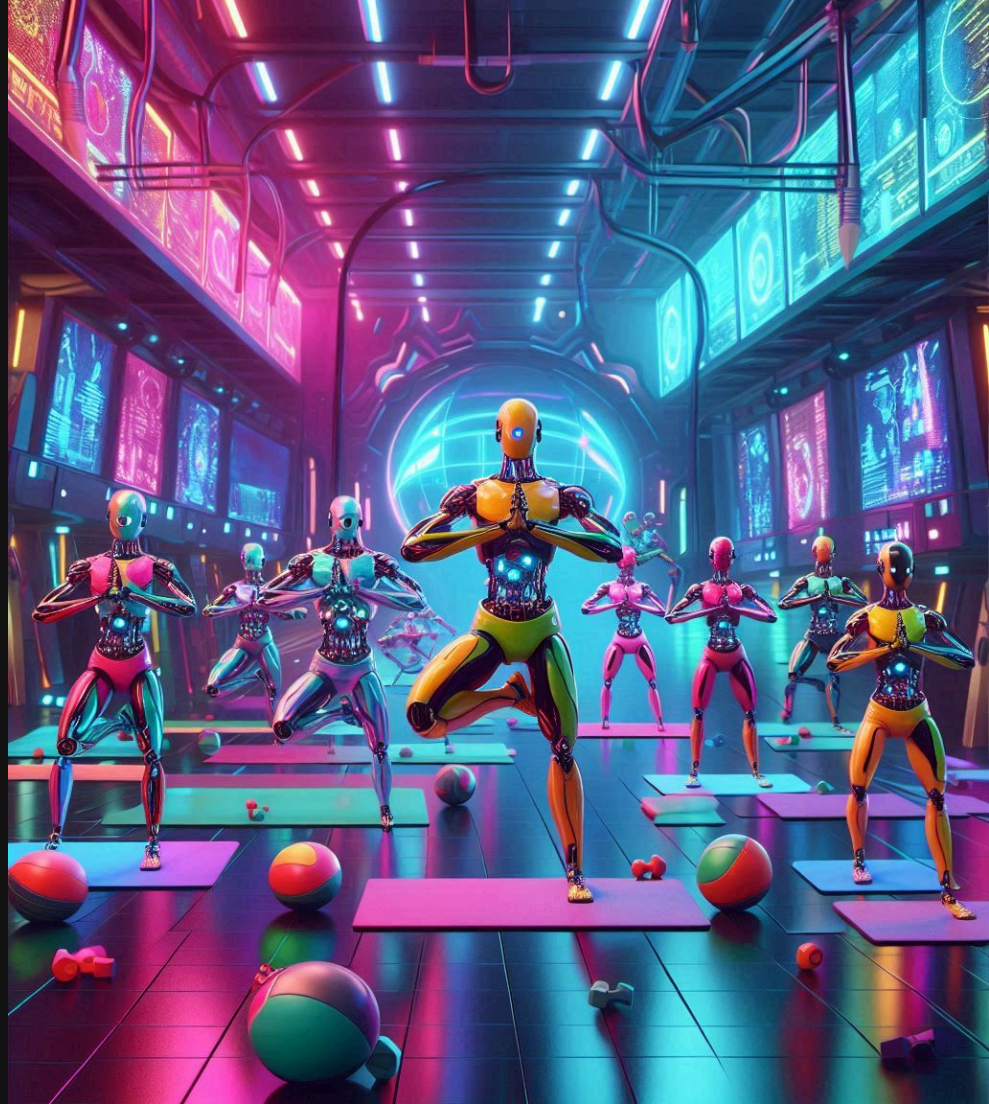
```
# These are optional elements. Feel free to remove any of them.
# status: "proposed | rejected | accepted | deprecated | … | superseded by ADR-0123"
# date: {YYYY-MM-DD when the decision was last updated}
# decision-makers: {list everyone involved in the decision}
# consulted: {list everyone whose 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}
```

Prompt 4 - Architecture Decision Records

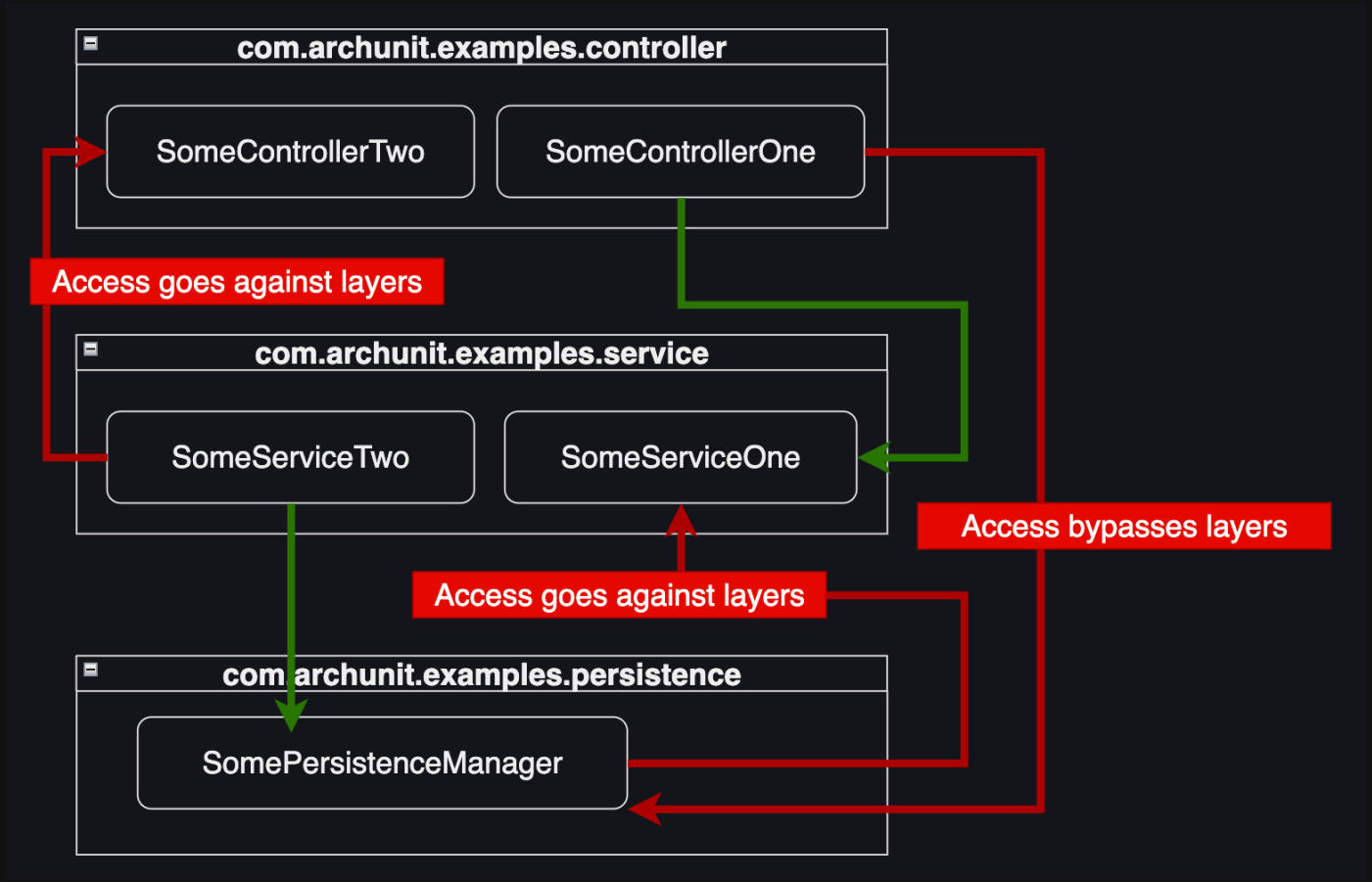
```
We are modifying our architecture to use Microsoft SQL Server as the database instead of the current solution.
Could you draft an Architecture Decision Record (ADR) explaining the rationale, alternatives considered,
and the implications of this decision?
Please format it in Markdown using the attached template file for consistency and clarity.
```

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);
```

Prompt 5 - Architecture Decision Records

```
I would like you to write ArchUnit unit tests for the architecture we just designed. The tests should enforce
key architectural rules, such as:

1. Ensuring specific dependencies between packages or layers.
2. Validating that services adhere to naming conventions.
3. Restricting certain components from accessing unauthorized layers or packages.

Please make the tests comprehensive, easy to extend, and aligned with Java best practices.
```

AI will not replace architects, but architects who use AI will replace those who don’t.

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