Architectual Views

Software Architecture

Richard Thomas

February 23, 2026

Interesting Software is Complex

Many aspects to the design of its architecture.

Architectural Design

Managing technical complexity.

Question

How do you describe a complex architecture, without making it too difficult to understand?

Question

How do you describe a complex architecture, without making it too difficult to understand?

Answer

Architectural Views

– Only consider one aspect at a time.

Architectural Views

- C4 Model *[Brown, 2023]*
  - context, structure, behaviour, infrastructure

Architectural Views

- C4 Model[Brown, 2023]
  - context, structure, behaviour, infrastructure
- 4+1 Views[Kruchten, 1995]
  - logical, process, development, physical, scenario

Architectural Views

- C4 Model[Brown, 2023]
  - context, structure, behaviour, infrastructure
- 4+1 Views[Kruchten, 1995]
  - logical, process, development, physical, scenario
- Software Architecture in Practice[Bass et al., 2021]
  - module, component-and-connector, allocation

Architectural Views

- C4 Model *[Brown, 2023]*
  - context, structure, behaviour, infrastructure
- 4+1 Views *[Kruchten, 1995]*
  - logical, process, development, physical, scenario
- Software Architecture in Practice *[Bass et al., 2021]*
  - module, component-and-connector, allocation
- NATO Architecture Framework *[Team, 2020]*
  - concepts, service, logical, physical resource, architecture foundation

Architectural Views

- C4 Model *[Brown, 2023]*
  - context, structure, behaviour, infrastructure
- 4+1 Views *[Kruchten, 1995]*
  - logical, process, development, physical, scenario
- Software Architecture in Practice *[Bass et al., 2021]*
  - module, component-and-connector, allocation
- NATO Architecture Framework *[Team, 2020]*
  - concepts, service, logical, physical resource, architecture foundation
- The Open Group Architecture Framework (TOGAF) *[Forum, 2022]*
- ISO/IEC/IEEE 42010:2011 *[iso, 2022]*

Diagrams & Notation

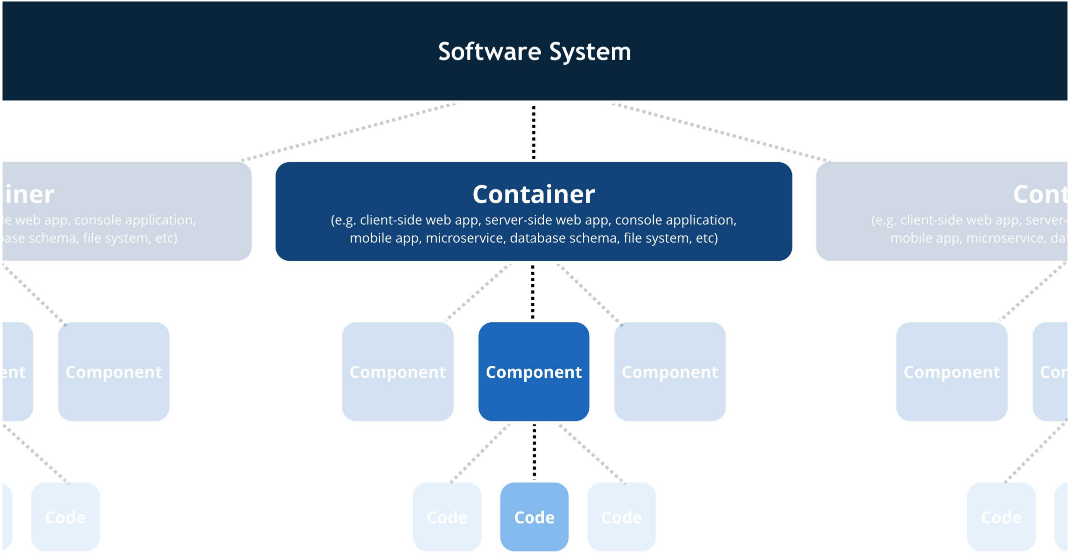
- A *good* diagram is worth a thousand words
    - A thousand diagrams is just confusing

Diagrams & Notation

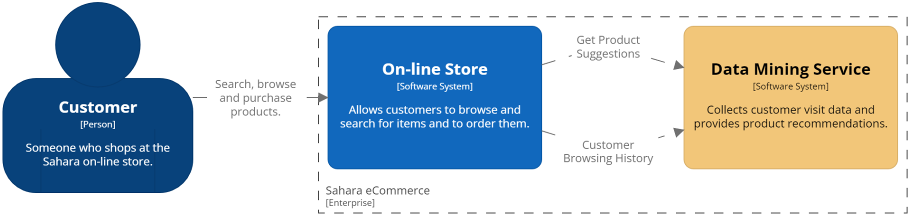
- A *good* diagram is worth a thousand words
  - A thousand diagrams is just confusing

- C4 – informal, simple structure *[Brown, 2023]*
- UML – formal, well-defined language *[uml, 2017]*
- You probably don't want to know about alternatives

C4 Model: Levels


Software System
iner
le web app, console application,
base schema, file system, etc)
Container
(e.g. client-side web app, server-side web app, console application,
mobile app, microservice, database schema, file system, etc)
Cont
(e.g. client-side web app, server-
mobile app, microservice, dat
ent
Component
Component
Component
Component
Component
Con
Code
Code
Code
Code
Code

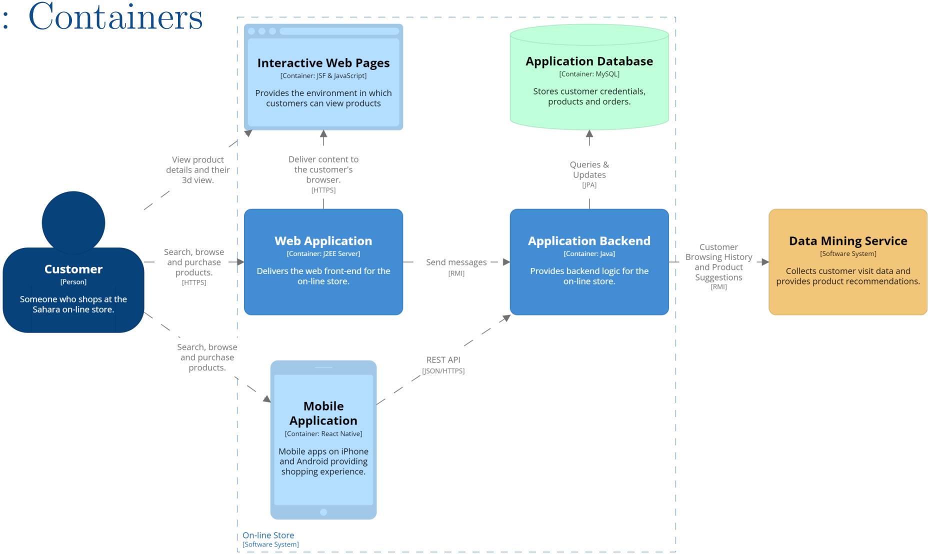
C4 Model: Context



Customer
[Person]
Someone who shops at the Sahara on-line store.
Search, browse and purchase products.
On-line Store
[Software System]
Allows customers to browse and search for items and to order them.
Get Product Suggestions
Customer Browsing History
Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.
Sahara eCommerce
[Enterprise]

How software system fits into broader *environment*

C4 Model: Containers



Interactive Web Pages
[Container: JSF & JavaScript]
Provides the environment in which customers can view products

Application Database
[Container: MySQL]
Stores customer credentials, products and orders.

View product details and their 3d view.

Deliver content to the customer's browser.
[HTTPS]

Queries & Updates
[JPA]

Customer
[Person]
Someone who shops at the Sahara on-line store.

Search, browse and purchase products.
[HTTPS]

Web Application
[Container: J2EE Server]
Delivers the web front-end for the on-line store.

Send messages
[RMI]

Application Backend
[Container: Java]
Provides backend logic for the on-line store.

Customer Browsing History and Product Suggestions
[RMI]

Data Mining Service
[Software System]
Collects customer visit data and provides product recommendations.

Search, browse and purchase products.

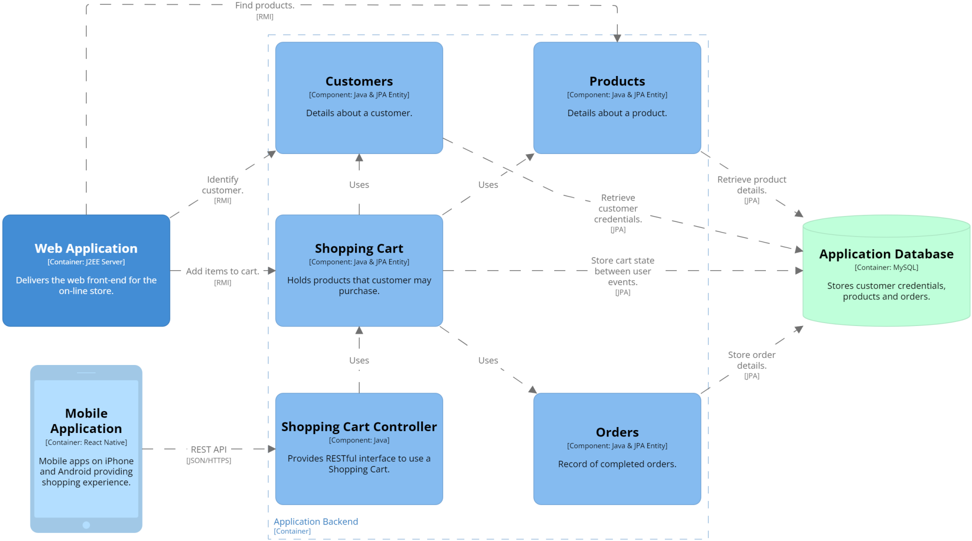
REST API
[JSON/HTTPS]

Mobile Application
[Container: React Native]
Mobile apps on iPhone and Android providing shopping experience.

On-line Store
[Software System]

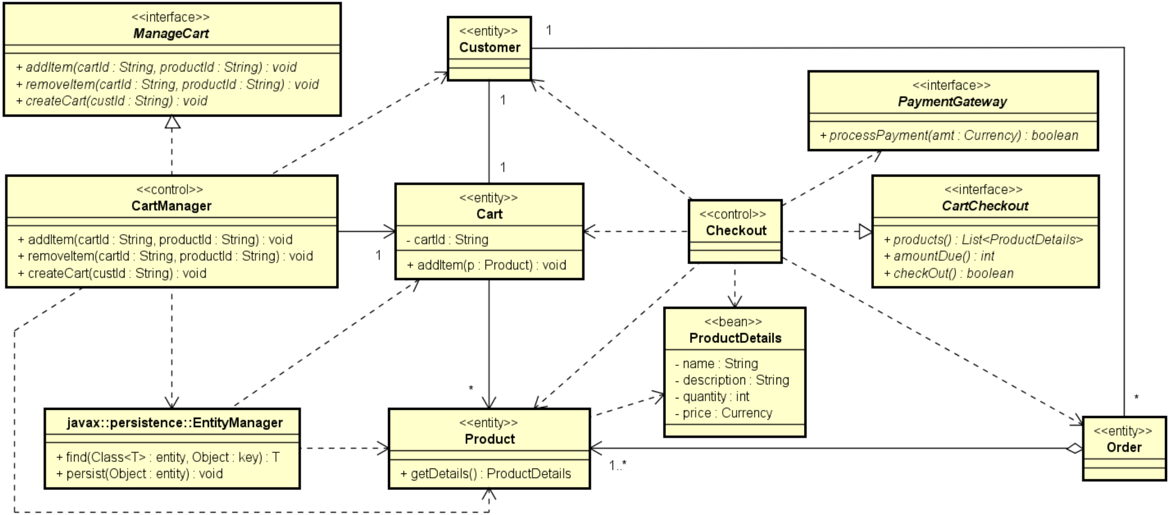
Structure of the software system

C4 Model: Components


Find products.
[RMI]
Customers
[Component: Java & JPA Entity]
Details about a customer.
Products
[Component: Java & JPA Entity]
Details about a product.
Identify customer.
[RMI]
Uses
Uses
Retrieve product details.
[JPA]
Retrieve customer credentials.
[JPA]
Web Application
[Container: J2EE Server]
Delivers the web front-end for the on-line store.
Add items to cart.
[RMI]
Shopping Cart
[Component: Java & JPA Entity]
Holds products that customer may purchase.
Store cart state between user events.
[JPA]
Application Database
[Container: MySQL]
Stores customer credentials, products and orders.
Uses
Uses
Store order details.
[JPA]
Mobile Application
[Container: React Native]
Mobile apps on iPhone and Android providing shopping experience.
REST API
[JSON/HTTPS]
Shopping Cart Controller
[Component: Java]
Provides RESTful interface to use a Shopping Cart.
Orders
[Component: Java & JPA Entity]
Record of completed orders.
Application Backend
[Container]

Elements that implement a container

C4 Model: Code



<<interface>>
ManageCart

+ addItem(cartId : String, productId : String) : void
+ removeItem(cartId : String, productId : String) : void
+ createCart(custId : String) : void

<<entity>>
Customer

<<interface>>
PaymentGateway

+ processPayment(amt : Currency) : boolean

<<control>>
CartManager

+ addItem(cartId : String, productId : String) : void
+ removeItem(cartId : String, productId : String) : void
+ createCart(custId : String) : void

<<entity>>
Cart

- cartId : String

+ addItem(p : Product) : void

<<control>>
Checkout

<<interface>>
CartCheckout

+ products() : List<ProductDetails>
+ amountDue() : int
+ checkOut() : boolean

<<bean>>
ProductDetails

- name : String
- description : String
- quantity : int
- price : Currency

javax::persistence::EntityManager

+ find(Class<T> : entity, Object : key) : T
+ persist(Object : entity) : void

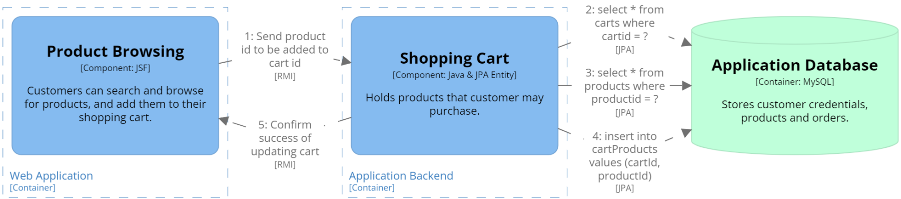
<<entity>>
Product

+ getDetails() : ProductDetails

<<entity>>
Order

Structure of code implementing a component

C4 Model: Dynamic



Product Browsing

[Component: JSF]

Customers can search and browse for products, and add them to their shopping cart.

Web Application

[Container]

1: Send product id to be added to cart id

[RMI]

5: Confirm success of updating cart

[RMI]

Shopping Cart

[Component: Java & JPA Entity]

Holds products that customer may purchase.

Application Backend

[Container]

2: select * from carts where cartid = ?

[JPA]

3: select * from products where productid = ?

[JPA]

4: insert into cartProducts values (cartId, productId)

[JPA]

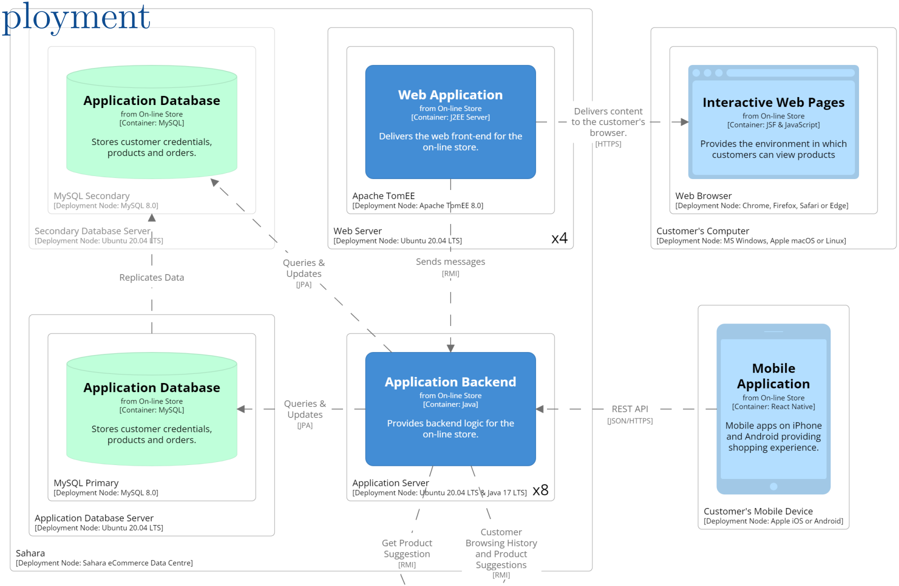
Application Database

[Container: MySQL]

Stores customer credentials, products and orders.

How parts of the model *collaborate* to deliver *behaviour*

C4 Model: Deployment



Application Database
from On-line Store
[Container: MySQL]

Stores customer credentials, products and orders.

MySQL Secondary
[Deployment Node: MySQL 8.0]

Secondary Database Server
[Deployment Node: Ubuntu 20.04 LTS]

Web Application
from On-line Store
[Container: J2EE Server]

Delivers the web front-end for the on-line store.

Apache TomEE
[Deployment Node: Apache TomEE 8.0]

Web Server
[Deployment Node: Ubuntu 20.04 LTS]

x4

Delivers content to the customer's browser.
[HTTPS]

Interactive Web Pages
from On-line Store
[Container: JSF & JavaScript]

Provides the environment in which customers can view products

Web Browser
[Deployment Node: Chrome, Firefox, Safari or Edge]

Customer's Computer
[Deployment Node: MS Windows, Apple macOS or Linux]

Replicates Data

Queries & Updates
[JPA]

Sends messages
[RMI]

Application Database
from On-line Store
[Container: MySQL]

Stores customer credentials, products and orders.

MySQL Primary
[Deployment Node: MySQL 8.0]

Application Database Server
[Deployment Node: Ubuntu 20.04 LTS]

Queries & Updates
[JPA]

Application Backend
from On-line Store
[Container: Java]

Provides backend logic for the on-line store.

Application Server
[Deployment Node: Ubuntu 20.04 LTS & Java 17 LTS]

x8

REST API
[JSON/HTTPS]

Mobile Application
from On-line Store
[Container: React Native]

Mobile apps on iPhone and Android providing shopping experience.

Customer's Mobile Device
[Deployment Node: Apple iOS or Android]

Get Product Suggestion
[RMI]

Customer Browsing History and Product Suggestions
[RMI]

Sahara
[Deployment Node: Sahara eCommerce Data Centre]

Infrastructure on which system will be deployed

4+1 Views

Logical – *Structure* of how the software is implemented

- components/classes, relationships, interactions

4+1 Views

Logical – *Structure* of how the software is implemented

- components/classes, relationships, interactions

Process – *Dynamic* behaviour

- concurrency & distribution, fault tolerance, process control, ...

4+1 Views

Logical – *Structure* of how the software is implemented

- components/classes, relationships, interactions

Process – *Dynamic* behaviour

- concurrency & distribution, fault tolerance, process control, ...

Development – *Organisation* of the software in the development environment

4+1 Views

Logical – *Structure* of how the software is implemented

- components/classes, relationships, interactions

Process – *Dynamic* behaviour

- concurrency & distribution, fault tolerance, process control, ...

Development – *Organisation* of the software in the development environment

Physical – *Map* executable software containers to hardware

- address non-functional requirements
  - availability, reliability, scalability, throughput, ...

4+1 Views

Logical – *Structure* of how the software is implemented

- components/classes, relationships, interactions

Process – *Dynamic* behaviour

- concurrency & distribution, fault tolerance, process control, ...

Development – *Organisation* of the software in the development environment

Physical – *Map* executable software containers to hardware

- address non-functional requirements
  - availability, reliability, scalability, throughput, ...

Scenario – *Demonstrate* functionality delivered by architecture

- use case details
  - *drive* functional design of architecture
  - *validate* design of architecture
  - *illustrate* purpose of architecture

Reading...

“Architectural Views” Notes[1] *[Thomas and Webb, 2023]*

[1] Remember, I said you had to read the notes.

References

[uml, 2017] (2017).
Unified Modeling Language.
OMG, 2.5.1 edition.
https://www.uml.org/.

[iso, 2022] (2022).
Software, Systems and Enterprise – Architecture Description (ISO/IEC/IEEE 42010:2022).
International Organization for Standardization.

[Bass et al., 2021] Bass, L., Clements, P., and Kazman, R. (2021).
Software Architecture in Practice.
Addison-Wesley, 4th edition.

[Brown, 2023] Brown, S. (2023).
The C4 Model for Visualising Software Architecture.
Leanpub.
https://leanpub.com/visualising-software-architecture.

[Forum, 2022] Forum, T. O. G. A. (2022).
The Open Group Architecture Framework Standard – Architecture Development Method.
The Open Group, 10 edition.
https://pubs.opengroup.org/togaf-standard/.

[Kruchten, 1995] Kruchten, P. (1995).
Architectural blueprints — the '4+1' view model of software architecture.
IEEE Software, 12(6):42–50.
https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf.

[Team, 2020] Team, A. C. (2020).
NATO Architecture Framework.
NATO, 4th edition.

[Thomas and Webb, 2023] Thomas, R. and Webb, B. (2023).
Architectural views.
`https://csse6400.uqcloud.net/handouts/views.pdf`.