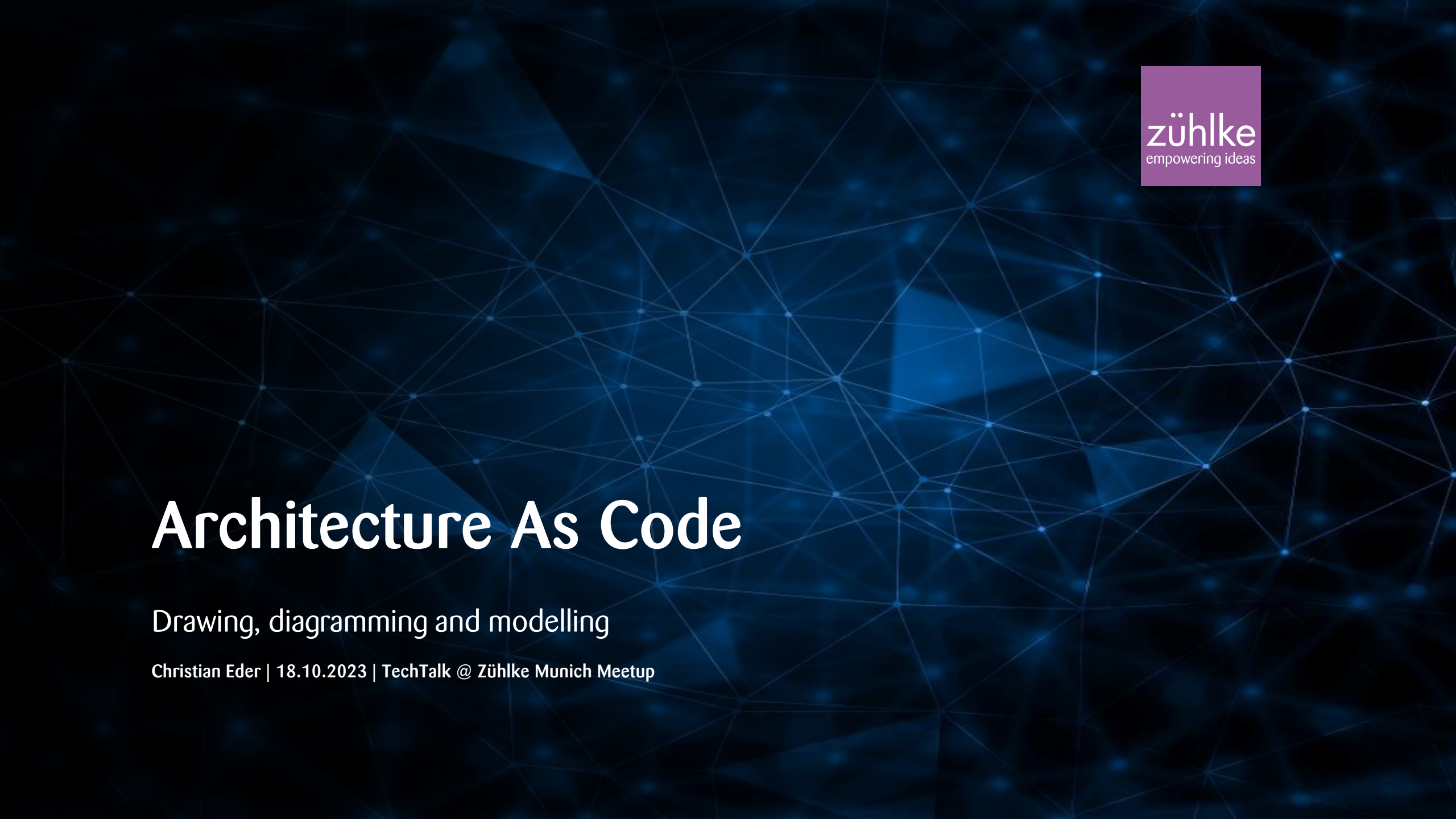


Architecture As Code

Drawing, diagramming and modelling

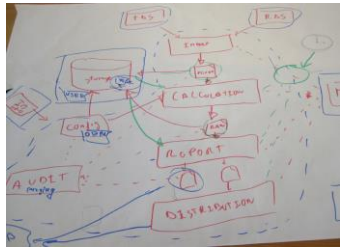
Christian Eder | 18.10.2023 | TechTalk @ Zühlke Munich Meetup

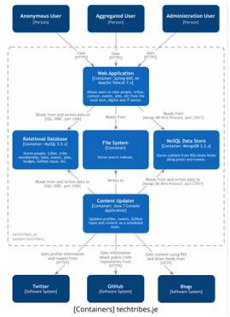
- What is architecture as code?
- Available tools
- Diagramming and modelling
- Drawing, configuring and coding

What is architecture as code?

Its about **documenting and describing** your **architecture**

Not primarily using drawings and diagrams





But tools closer to the ones used by developers

Some nomenclature used in this talk

Drawing is process of **manually creating diagrams**
using tools such as Visio, PowerPoint, diagrams.net, whiteboards…

Modelling comes before drawing and is typically **broader**:
you **don't draw every aspect** of your architectural model

Diagramming is the process of manually or automatically **drawing diagrams of a model**

… **as configuration** is about expressing things using **configuration languages** such as YAML

… **as code** is about expressing things using general purpose **programming languages**

Available tools for diagramming

Embeddable into markdown files on GitHub

Mermaid

https://mermaid-js.github.io/mermaid

PlantUML

https://plantuml.com

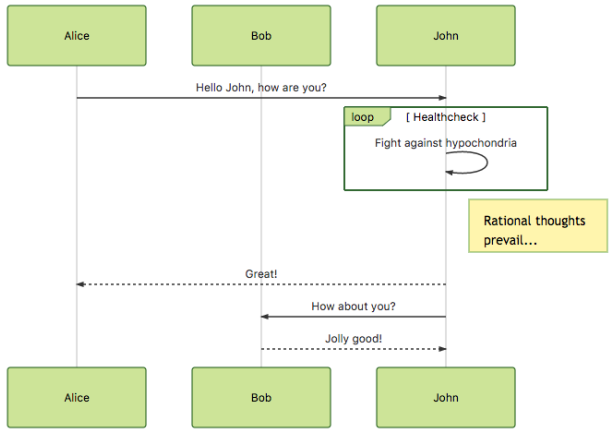
Renders nice interactive diagrams

Ilograph

https://www.ilograph.com

All of these are tools to convert **textual descriptions** into **visual diagrams**

```
sequenceDiagram
    participant Alice
    participant Bob
    Alice->>John: Hello John, how are you?
    loop Healthcheck
        John->>John: Fight against hypochondria
    end
    Note right of John: Rational thoughts <br/>prevail!
    John-->>Alice: Great!
    John->>Bob: How about you?
    Bob-->>John: Jolly good!
```



Eder, Christian

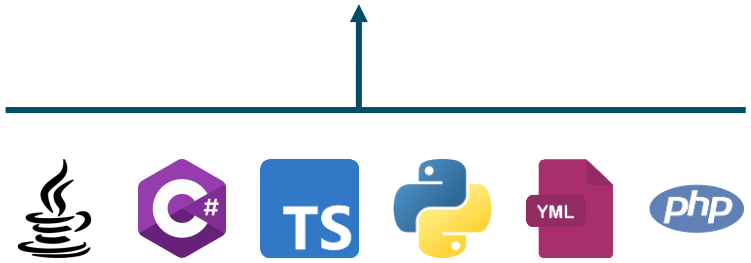
Modelling and diagramming as code: Structurizr

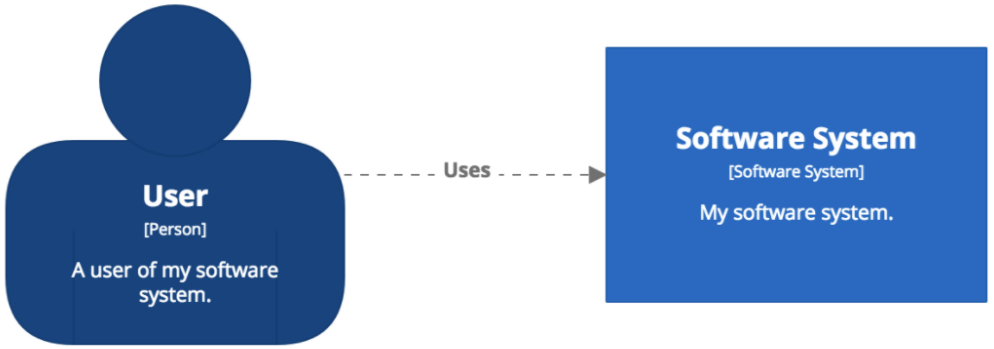
https://github.com/ChristianEder/architecture-as-code/blob/1ab6ad883a50e213f989a27a11c4b1f5a647f755/available-tools/structurizr/src/index.ts

Model

Diagram

```
Workspace workspace = new Workspace("Getting Started", "This is a model of my software system.");
Model model = workspace.getModel();

Person user = model.addPerson("User", "A user of my software system.");
SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
user.uses(softwareSystem, "Uses");

ViewSet views = workspace.getViews();
SystemContextView contextView = views.createSystemContextView(softwareSystem, "SystemContext", "An example of a
contextView.addAllSoftwareSystems();
contextView.addAllPeople();
```





Modelling and diagramming as code: Structurizr

Modelling your architecture using general purpose programming languages…

```
export function defineModel() {
    const workspace = new Workspace('Architecture as code example workspace', 'Describes the architecture of a fictious IoT system');
    workspace.model.impliedRelationshipsStrategy = new CreateTaggedImpliedRelationshipsUnlessAnyRelationshipExistsStrategy();

    const user = workspace.model.addPerson('User', 'Users of the system')!;

    const cloud = defineCloudSystem(workspace.model, user);
    const onPremiseSystem = defineOnPremiseSystem(workspace.model, cloud.containers.machineMetadataTransferStorage);
    const factoryFloor = defineFactoryFloorSystem(workspace.model, cloud.containers.iotHub, cloud.containers.dps);
    const gatewayProvisioningEnvironment = defineGatewayProvisioningEnvironmentSystem(workspace.model, factoryFloor.gateway, cloud.co
```

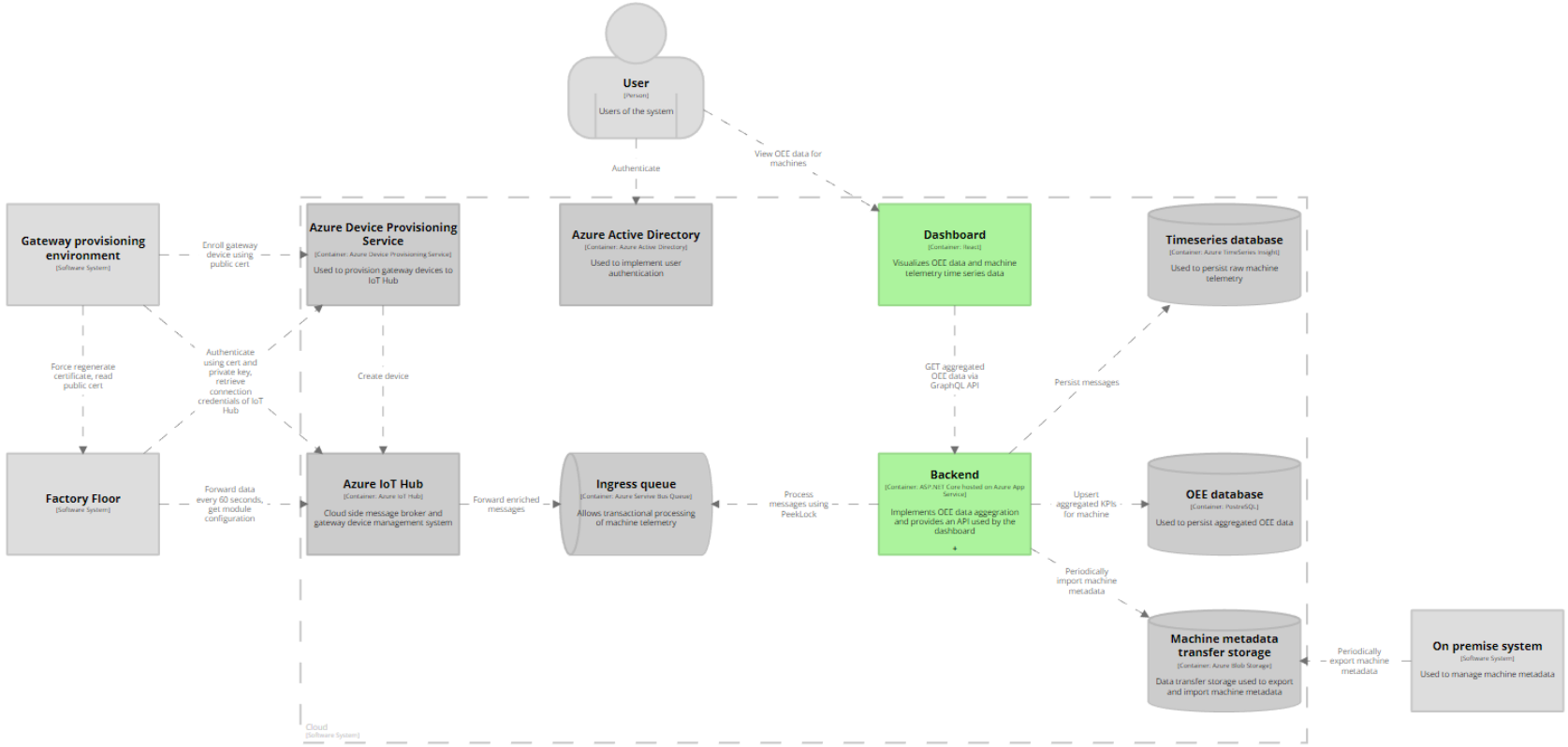
… giving you the ability to structure your architecture description the same way you'd structure "normal code"…

```
export function defineOnPremiseSystem(model: Model, machineMetadataTransferStorage: Container) {
    const system = model.addSoftwareSystem('On premise system', 'Used to manage machine metadata')!;

    system.uses(machineMetadataTransferStorage, 'Periodically export machine metadata');

    return { system };
}
```

Modelling and diagramming as code: Structurizr

… and the ability to easily generate diagrams from that architecture model

```
export function defineOverviewDiagram(workspace: Workspace, cloudSystem: SoftwareSystem) {
    const diagram = workspace.views.createContainerView(cloudSystem, 'Overview', 'Shows all related components');

    diagram.addAllSoftwareSystems();
    diagram.addAllContainers();
    diagram.addAllPeople();
}
```



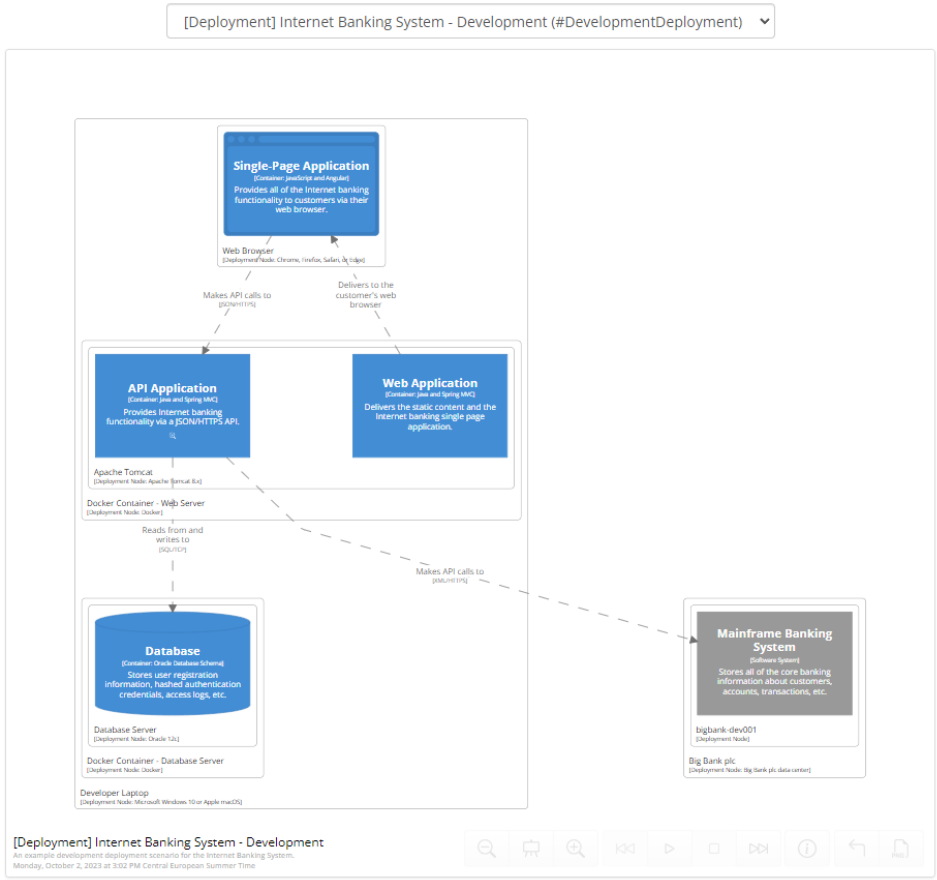
Structurizr DSL

Great to get started – or for simple use cases
https://structurizr.com/dsl

```
 * - "Big Bank plc - Internet Banking System" (https://structurizr.com/share/36141/)
*/
workspace "Big Bank plc" "This is an example workspace to illustrate the key features of Structurizr, via the DSL, based around a fictional online banking system." {

    model {
        customer = person "Personal Banking Customer" "A customer of the bank, with personal bank accounts." "Customer"

        group "Big Bank plc" {}

        # relationships between people and software systems
        customer -> internetBankingSystem "Views account balances, and makes payments using"
        internetBankingSystem -> mainframe "Gets account information from, and makes payments using"
        internetBankingSystem -> email "Sends e-mail using"
        email -> customer "Sends e-mails to"
        customer -> supportStaff "Asks questions to" "Telephone"
        supportStaff -> mainframe "Uses"
        customer -> atm "Withdraws cash using"
        atm -> mainframe "Uses"
        backoffice -> mainframe "Uses"

        # relationships to/from containers
        customer -> webApplication "Visits bigbank.com/ib using" "HTTPS"
        customer -> singlePageApplication "Views account balances, and makes payments using"
        customer -> mobileApp "Views account balances, and makes payments using"
        webApplication -> singlePageApplication "Delivers to the customer's web browser"

        # relationships to/from components
        singlePageApplication -> signinController "Makes API calls to" "JSON/HTTPS"
        singlePageApplication -> accountsSummaryController "Makes API calls to" "JSON/HTTPS"
        singlePageApplication -> resetPasswordController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> signinController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> accountsSummaryController "Makes API calls to" "JSON/HTTPS"
        mobileApp -> resetPasswordController "Makes API calls to" "JSON/HTTPS"
        signinController -> securityComponent "Uses"
        accountsSummaryController -> mainframeBankingSystemFacade "Uses"
        resetPasswordController -> securityComponent "Uses"
        resetPasswordController -> emailComponent "Uses"
        securityComponent -> database "Reads from and writes to" "SQL/TCP"
        mainframeBankingSystemFacade -> mainframe "Makes API calls to" "XML/HTTPS"
        emailComponent -> email "Sends e-mail using"

        deployment Environment "Development" {}

        deployment Environment "Live" {}
    }

    views {
        systemlandscape "SystemLandscape" {
            include *
            autoLayout
        }

        systemcontext internetBankingSystem "SystemContext" {
            include *
            animation {
                internetBankingSystem
                customer
                mainframe
                email
```

Structurizr DSL v1.32.0 - some features (e.g. `!docs`, `!adrs`, `!script`, etc) are unavailable via this demo page; see Help - DSL for details.

DSL language reference | DSL cookbook | Examples: Getting started | Big Bank plc | Microservices | Amazon Web Services

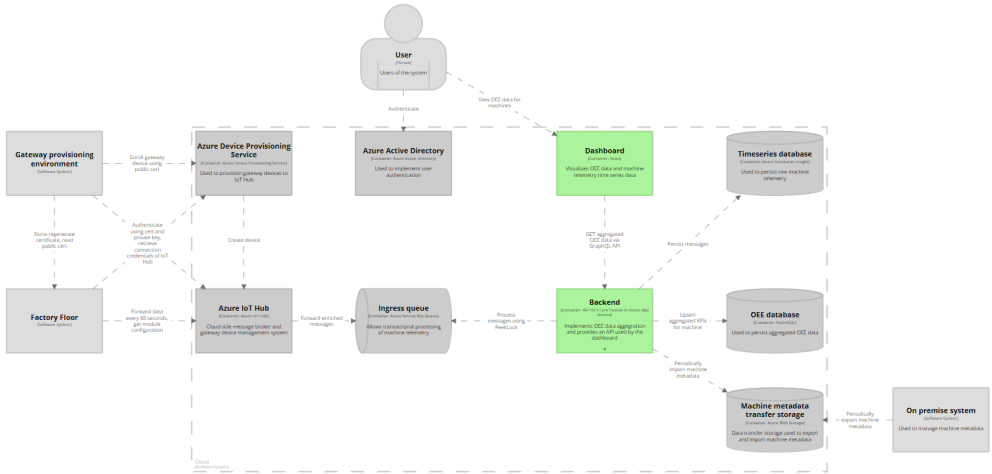


Benefits of modelling over diagramming

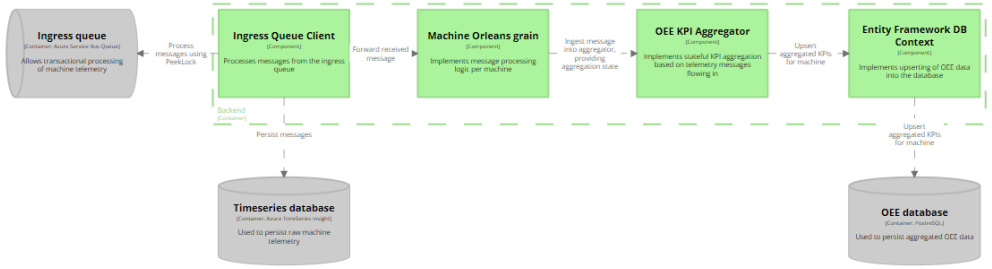
Benefits of modelling over diagramming

The ability to easily create different **consistent views** on the **same model**

```
export function defineOverviewDiagram(workspace: Workspace, cloudSystem: SoftwareSystem) {
    const diagram = workspace.views.createContainerView(cloudSystem, 'Overview', 'Shows all

    diagram.addAllSoftwareSystems();
    diagram.addAllContainers();
    diagram.addAllPeople();
}
```



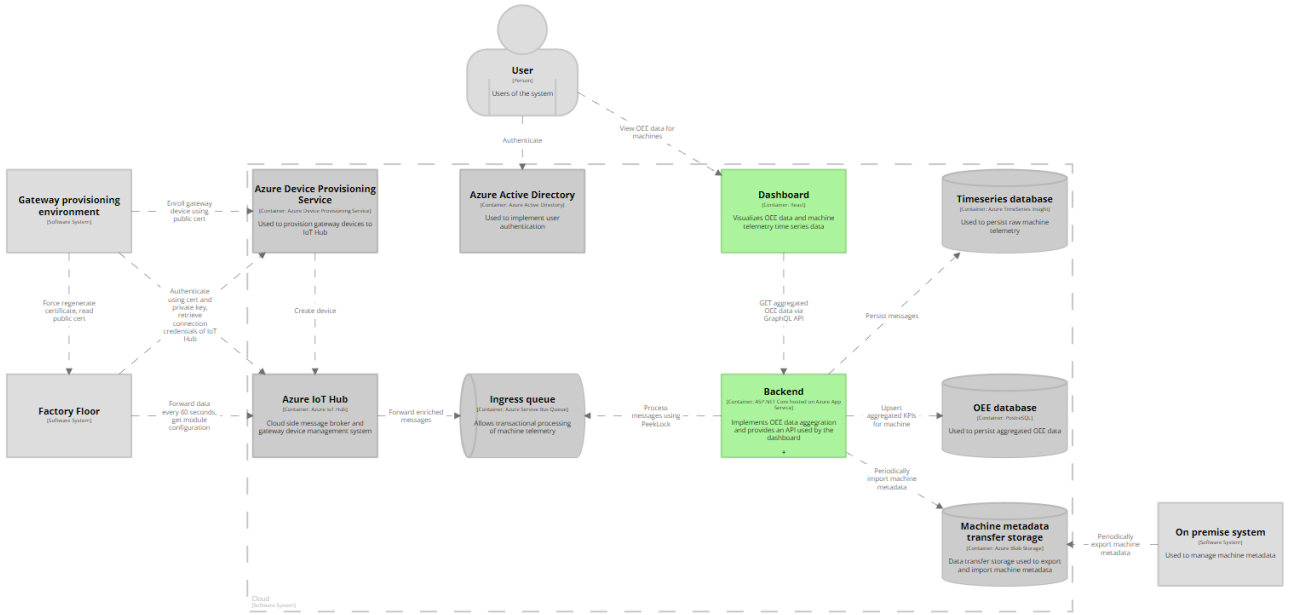

```
export function defineDataIngressDiagram(workspace: Workspace, backend: Container) {
    const diagram = workspace.views.createComponentView(backend, 'Data Ingress', 'Shows

    workspace.model.softwareSystems.filter(s => s.tags.contains('data-ingress')).forEach

    backend.components.filter(c => c.tags.contains('data-ingress')).forEach(c => {
        diagram.addComponent(c);
        diagram.addNearestNeighbours(c);
    });
}
```



Benefits of modelling over diagramming

The ability to consistently (re-) **style your diagrams**

```
export function defineStyling(workspace: Workspace) {
    renderTagAsShape(workspace, Tags.Person, Shape.Person);
    renderTagAsShape(workspace, 'queue', Shape.Pipe);
    renderTagAsShape(workspace, 'database', Shape.Cylinder);

    renderTagWithColor(workspace, Tags.Container, "#ACF39D");
    renderTagWithColor(workspace, Tags.Component, "#ACF39D");
    renderTagWithColor(workspace, 'azure-component', "#CCCCCC");
}
```



Benefits of coding over drawing

Benefits of coding over drawing

Easier **versioning** of your documentation

Minor layout fixes

main

ChristianEder-z committed 3 minutes ago

Showing **1 changed file** with **1 addition** and **1 deletion**.

2 architecture.azure.drawio

```
:VJ1XCJL7K1rQryf3nwr42K9oeHjgSayA9tW0WNkaPsovtYAi2eW65b1ooU15mCs6VCnqAvGtXrg0RXZ++I8tq2KUBGNdo
'JxvClqMhdJ1XGKrLC3rgnJfvbwrYxL9IZE9weSyg5om0WPsant4+RevWi2ypo8163qRQtrrMFY06FOUReMa/XAJyu8D/h
```

Add timeseries database

main

ChristianEder-z committed 28 seconds ago

Showing **1 changed file** with **9 additions** and **2 deletions**.

11 available-tools/structurizr/src/model/cloud/cloudSystem.ts

```
@@ -22,6 +22,11 @@ export function defineCloudSystem(model: Model, user: Person) {
22 22      oeeDatabase.tags.add('data-ingress');
23 23      oeeDatabase.tags.add('azure-component');
24 24
   25 +    const timeseriesDatabase = system.addContainer('Timeseries database', 'Used to persist raw mac
   26 +    timeseriesDatabase.tags.add('database');
   27 +    timeseriesDatabase.tags.add('data-ingress');
   28 +    timeseriesDatabase.tags.add('azure-component');
   29 +
25 30      const activeDirectory = system.addContainer('Azure Active Directory', 'Used to implement user
26 31      activeDirectory.tags.add('azure-component');
27 32
@@ -31,7 +36,7 @@ export function defineCloudSystem(model: Model, user: Person) {
31 36
32 37      const dashboard = system.addContainer('Dashboard', 'Visualizes OEE data and machine telemetry
33 38
34    -    const backend = defineBackendContainer(system, ingressQueue, oeeDatabase, machineMetadataTrans
   39 +    const backend = defineBackendContainer(system, ingressQueue, oeeDatabase, timeseriesDatabase,
```

Benefits of coding over drawing

Added **interoperability** between tools

A Structurizr model

```
export function defineModel() {
    const workspace = new Workspace('Architecture as code example workspace', 'Describes the architecture of a fictious IoT system');
    workspace.model.impliedRelationshipsStrategy = new CreateTaggedImpliedRelationshipsUnlessAnyRelationshipExistsStrategy();

    const user = workspace.model.addPerson('User', 'Users of the system')!;

    const cloud = defineCloudSystem(workspace.model, user);
    const onPremiseSystem = defineOnPremiseSystem(workspace.model, cloud.containers.machineMetadataTransferStorage);
    const factoryFloor = defineFactoryFloorSystem(workspace.model, cloud.containers.iotHub, cloud.containers.dps);
    const gatewayProvisioningEnvironment = defineGatewayProvisioningEnvironmentSystem(workspace.model, factoryFloor.gateway, cloud.co
```

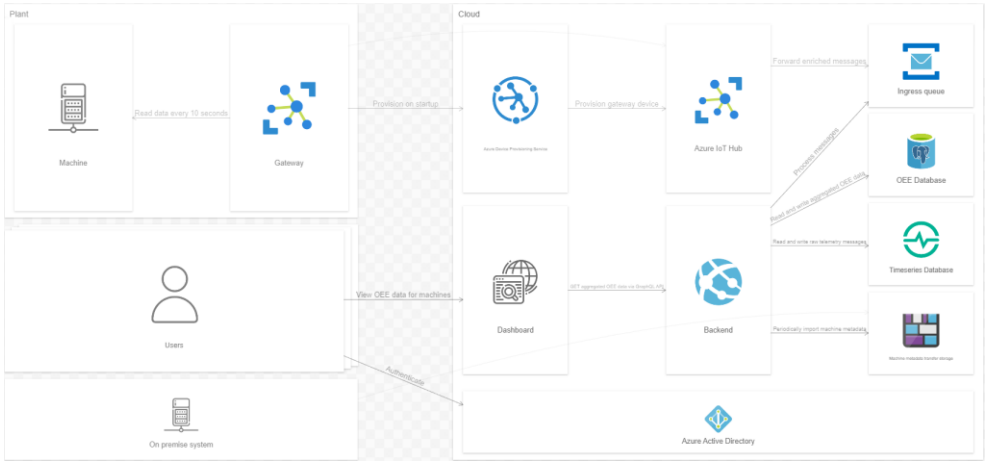
+

A few lines of conversion code

```
const overviewPerspective = new IlographPerspective('Overview');
structurizrModel.workspace.model.relationships
    .filter(r => r.source.type === Container.type && r.destination.type === Container.type)
    .forEach(r => {
        overviewPerspective.addRelation(new IlographRelation(r.source.name, r.destination.name, r.description));
});
ilographWorkspace.addPerspective(overviewPerspective);
```

=

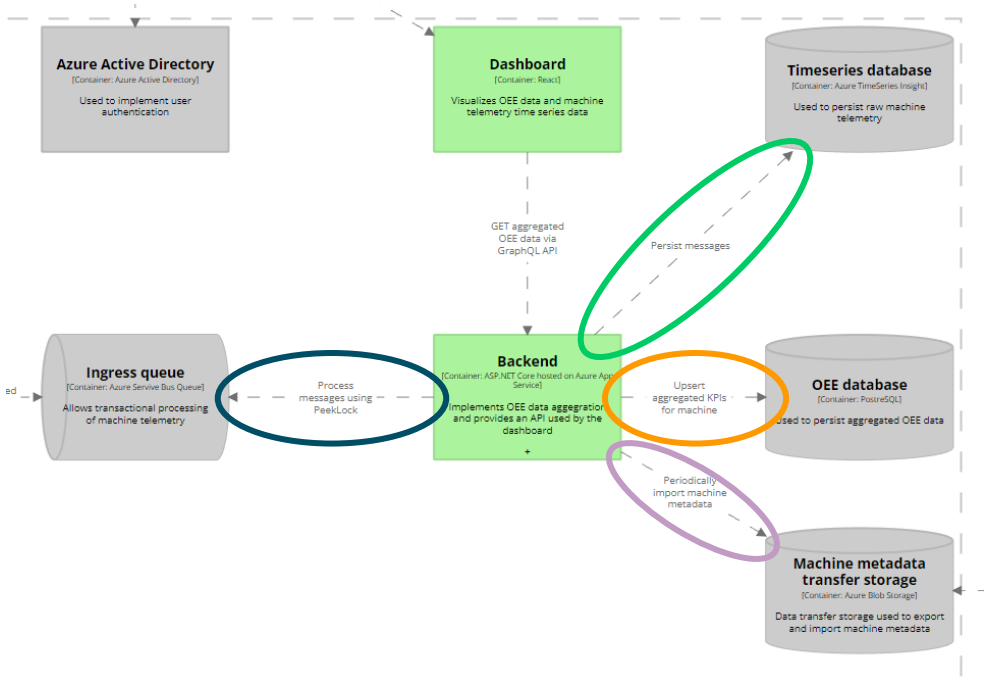
An Ilograph diagram



Benefits of coding over drawing

Component **dependencies** show up in **diagram** and **code** consistently

```
const backend = defineBackendContainer(system, ingressQueue, oeeDatabase, timeseriesDatabase, machineMetadataTransferStorage);
```



Benefits of coding over drawing

Increased chances of your documentation staying **up to date**

Get in touch!

christian.eder@zuehlke.com

https://github.com/ChristianEder/architecture-as-code

https://medium.com/@christian.johann.eder

@_ceder

zühlke
empowering ideas

… or right now, at the 🍕

Innovation Talk: Embedded DevOps meets Sustainability

Hosted By
Christian E.

Organisatoren-Tools

Zühlke Munich
Public group
(111)

Mittwoch, 15. November 2023 um 17:30 bis Mittwoch, 15. November 2023 um 20:30 CET
Zum Kalender hinzufügen

Zühlke Engineering GmbH
St.-Martin-Straße 102 · München

So findest du uns
Im Gebäude in den ersten Stock, dort

15. NOV. 2023 17:30 CET
Innovation Talk: Embedded DevOps meets Sustainability

Du nimmst teil!
RSVP bearbeiten

Teilen