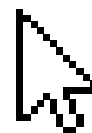




Diagramming in Software engineering: SE Radio episode 566



Gonzalo Alonso Fernández **U0282104**

Raúl Mera Soto **U0287827**

Laura Gómez Menéndez **U0275725**

Daniel Sinne Argüelles **U0282500**

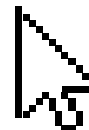


**SOFTWARE
ARCHITECTURE**



Table of contents

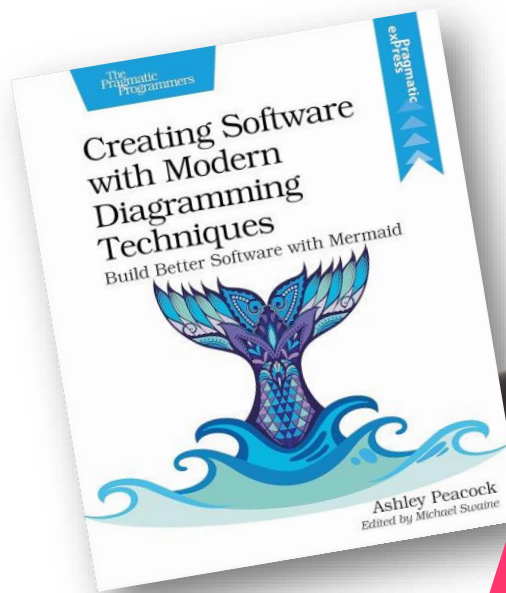
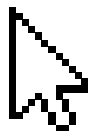
- Introduction
- Why is diagramming important
- Ashley's opinion about UML
- Types of diagrams in software engineering
- Tools for modern diagramming techniques
 - Draw.io
 - PlantUML
 - Mermaid
- C4 model
- Other useful tools
 - Miro
 - UML Lab





**SOFTWARE
ENGINEERING
RADIO** PODCAST

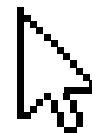
Introduction





Why is diagramming important?

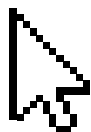
- Make information digestible, providing a visual and easy-to-understand representation.
- Essential for documenting architectures and explaining complex concepts.
- Can be applied in the whole software life cycle.





Ashley's opinion about UML

- UML's poor reputation.
- Emphasis on Notation over Diagram Power.
- Advancements in diagramming tools.
- Lack of Awareness and Educational Resources.

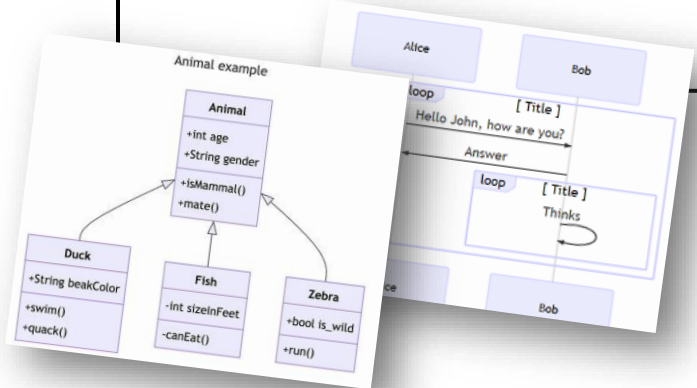




Types of Diagrams in Software Engineering

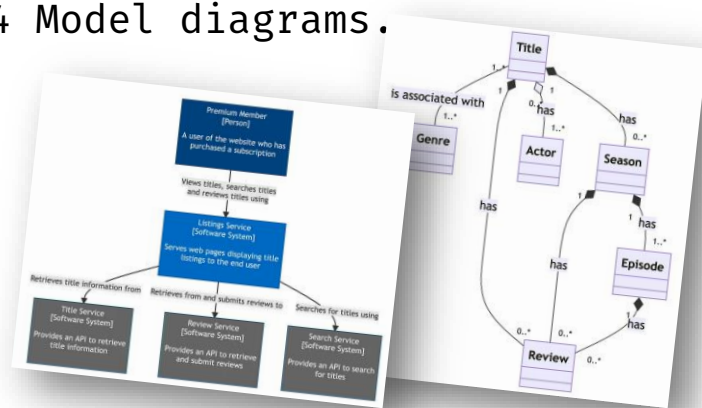
Short-lived diagrams

- Class diagrams.
- Sequence diagrams.



Long-lived diagrams

- Domain Model diagrams.
- C4 Model diagrams.





Tools for modern diagramming techniques

Draw.io

Ease of use



PlantUML

Diagrams as code



Mermaid

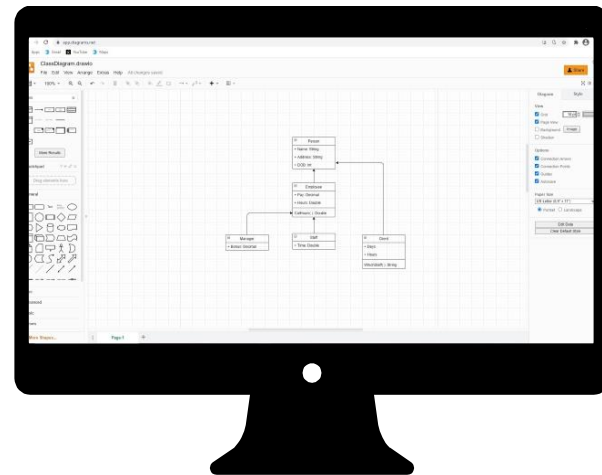
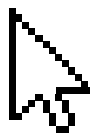
Browser rendering





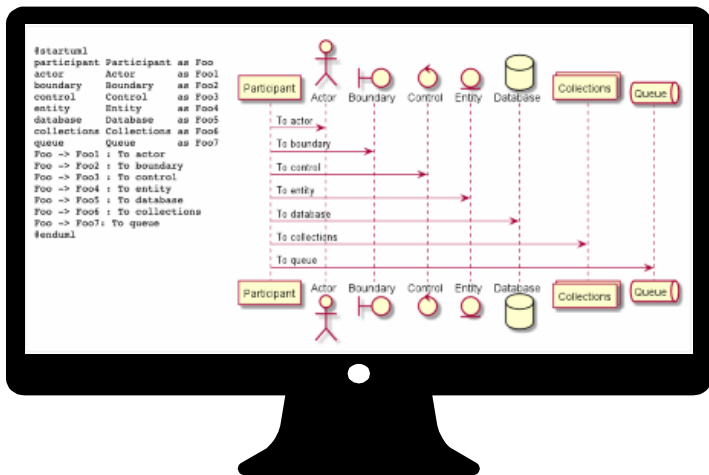
Draw.io

- + Open source
- + Easy to use (Drag & drop)
- Slow to change
- Maintenance nightmare
- Unreadable format
- Bad option for version control

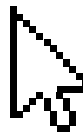




PlantUML



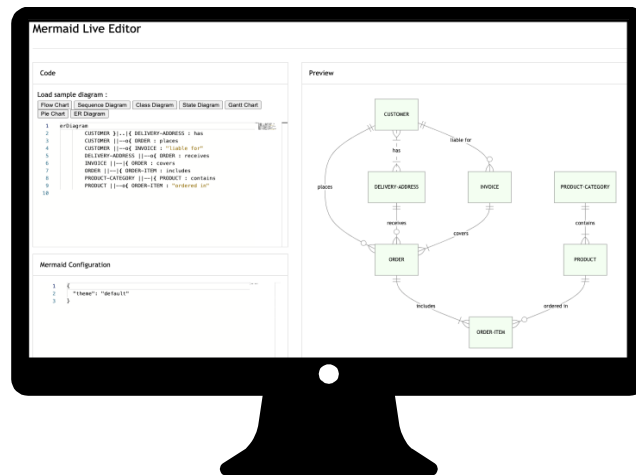
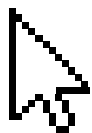
- + Open source
- + Diagrams as code
- + Text-based
 - + Fast creation and editing
 - + Consistency
 - + Version control friendly
- Needs Java





Mermaid

- + Open source
- + Similar to PlantUML
 - + JavaScript
 - + No separate diagram tool
 - + Browser rendering
 - + Easier version control





Mermaid syntax

Mermaid Live Editor

THEME SECURITY DOCUMENTATION TUTORIAL MERMAID CLI

Mermaid `</> Code` `Config` Auto sync `DOCS` Diagram Pan & Zoom `FULL SCREEN` `SAVE TO MERMAID CHART`

```
1 classDiagram
2   Animal <|-- Duck
3   Animal <|-- Fish
4   Animal <|-- Zebra
5   Animal : +int age
6   Animal : +String gender
7   Animal : +isMammal()
8   Animal : +mate()
9   class Duck{
10    +String beakColor
11    +swim()
12    +quack()
13  }
14   class Fish{
15    -int sizeInFeet
16    -canEat()
17  }
18   class Zebra{
19    +bool is_wild
20    +run()
21  }
```

> Sample Diagrams

> History `↓` `↑` `↺` `↻`

> Actions

```
classDiagram
    Animal <|-- Duck
    Animal <|-- Fish
    Animal <|-- Zebra
    Animal : +int age
    Animal : +String gender
    Animal : +isMammal()
    Animal : +mate()
    class Duck {
        +String beakColor
        +swim()
        +quack()
    }
    class Fish {
        -int sizeInFeet
        -canEat()
    }
    class Zebra {
        +bool is_wild
        +run()
    }
```





Mermaid styling

Mermaid Live Editor

THEME SECURITY DOCUMENTATION TUTORIAL MERMAID CLI

Code Config Auto sync DOCS Diagram Pan & Zoom FULL SCREEN SAVE TO MERMAID CHART

```
1 {
2   "theme": "base",
3   "themeVariables": {
4     "primaryColor": "#011013",
5     "primaryTextColor": "#fff",
6     "primaryBorderColor": "#10f3c0",
7     "lineColor": "#10f3a0",
8     "secondaryColor": "#000100",
9     "tertiaryColor": "#fff"
10  }
11 }
```

Sample Diagrams

History

Actions

```
classDiagram
    class Animal {
        +int age
        +String gender
        +isMammal()
        +mate()
    }
    class Duck {
        +String beakColor
        +swim()
        +quack()
    }
    class Fish {
        -int sizeInFeet
        -canEat()
    }
    class Zebra {
        +bool is_wild
        +run()
    }
    Animal <|-- Duck
    Animal <|-- Fish
    Animal <|-- Zebra
```





Mermaid Sequence Diagrams

Handle the layout and the rendering for you

Lifelines

Individual nodes
the messages are
passing between

Flowcharts

Square boxes,
cylinders,
diamonds...

Dotted lineback

To show
responses

Solid line

To show
requests

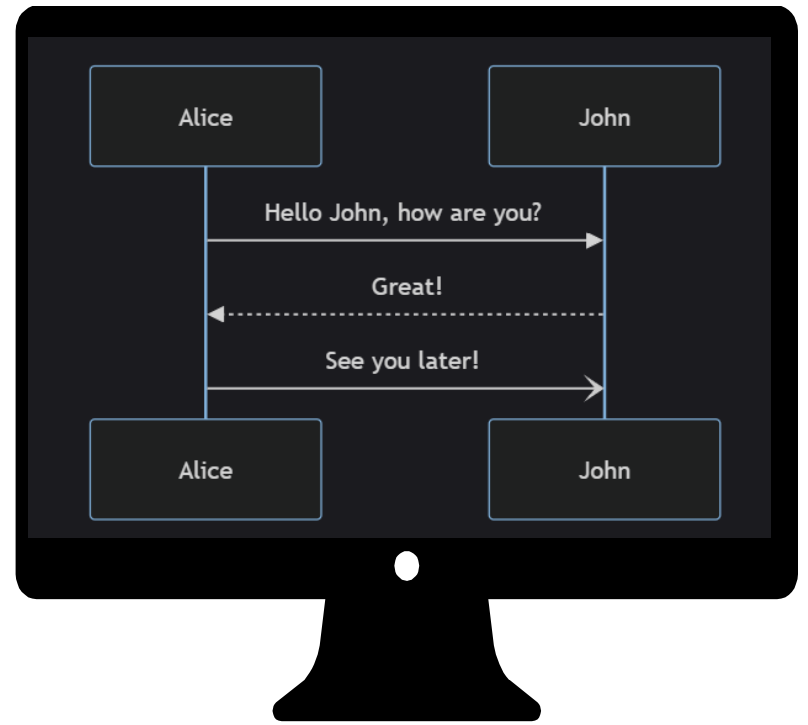
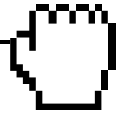


Mermaid Sequence Diagrams



Participants

```
sequenceDiagram
  Alice->>John: Hello John,how are you?
  John-->>Alice: Great!
  Alice->>John: See you later!
```



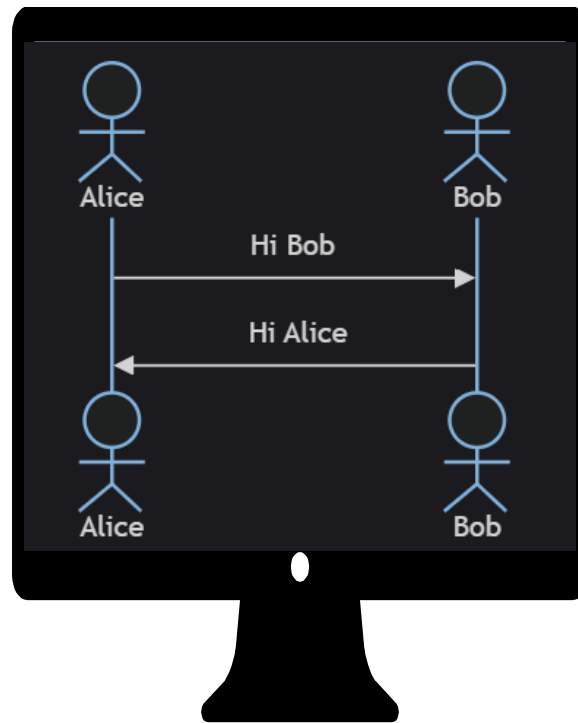


Mermaid Sequence Diagrams



Actors

```
sequenceDiagram
  actor Alice
  actor Bob
  Alice->>Bob: Hi Bob
  Bob->>Alice: Hi Alice
```

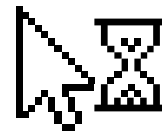




C4 Model

Simon Brown created this way to model your software Architecture.

The name C4 Model comes from **4 diagrams**.





C4 Model

Diagrams



System Context Diagram

Highest level of abstraction and provides an overview of the system



Component Diagram

Explores deep the internal structure of each container



Container View

Show **how the containers interact** with each other and with external systems or users



Code diagram

Provides a **detailed view** of the code-level elements within the components



Component vs Container

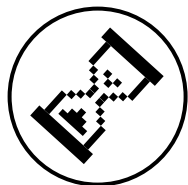
Represent different architectural elements in a system

Component

Specific software module

Container

Higher-level grouping or environment that contains multiple components or other containers.



Other Useful Tools



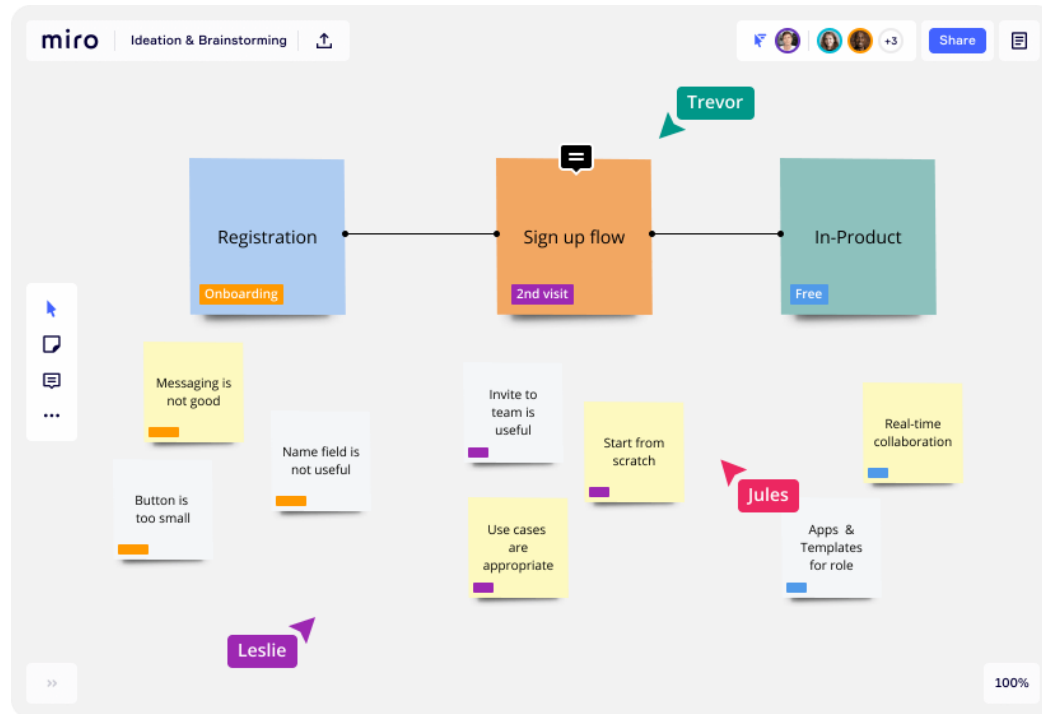
miro





- Digital whiteboard
- Used for brainstorming and event storming (Domain-Driven Design)
- Work remotely

Miro

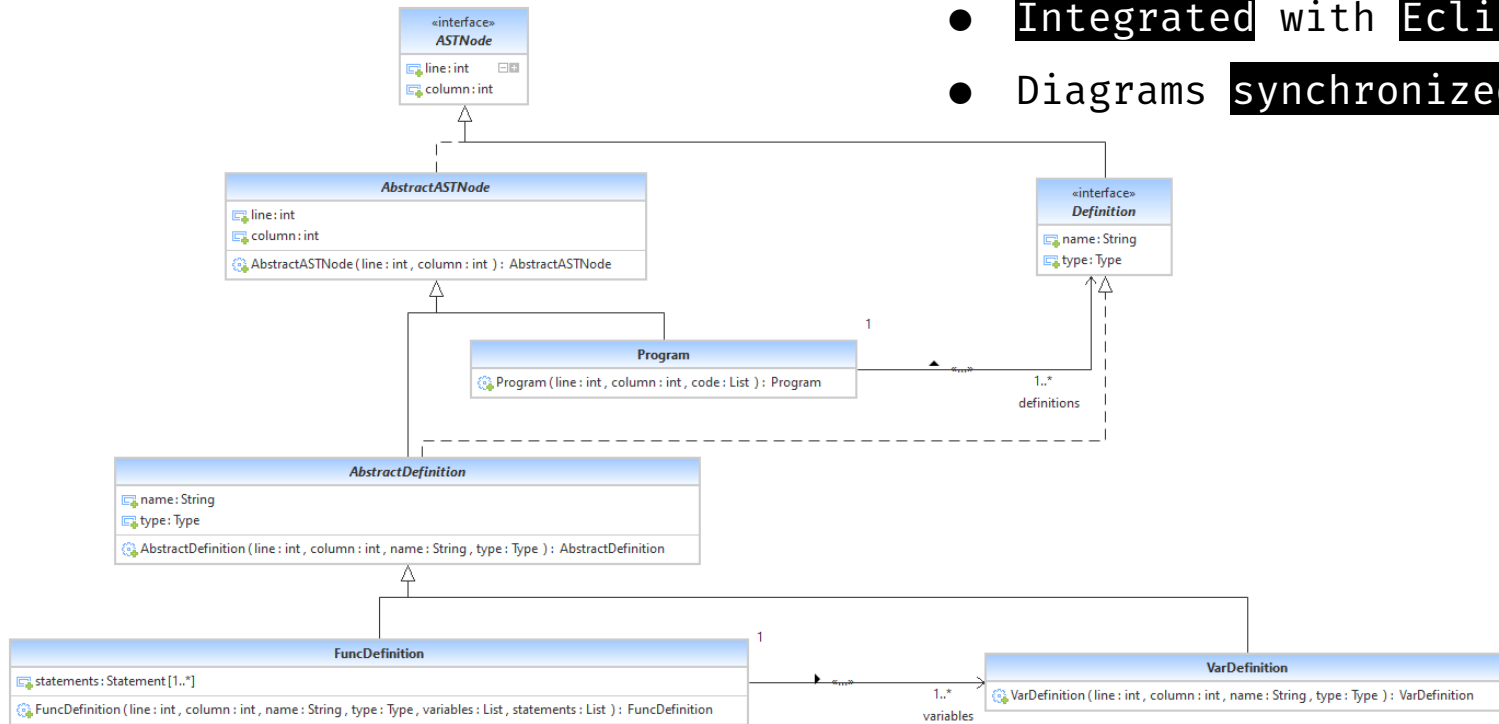


miro

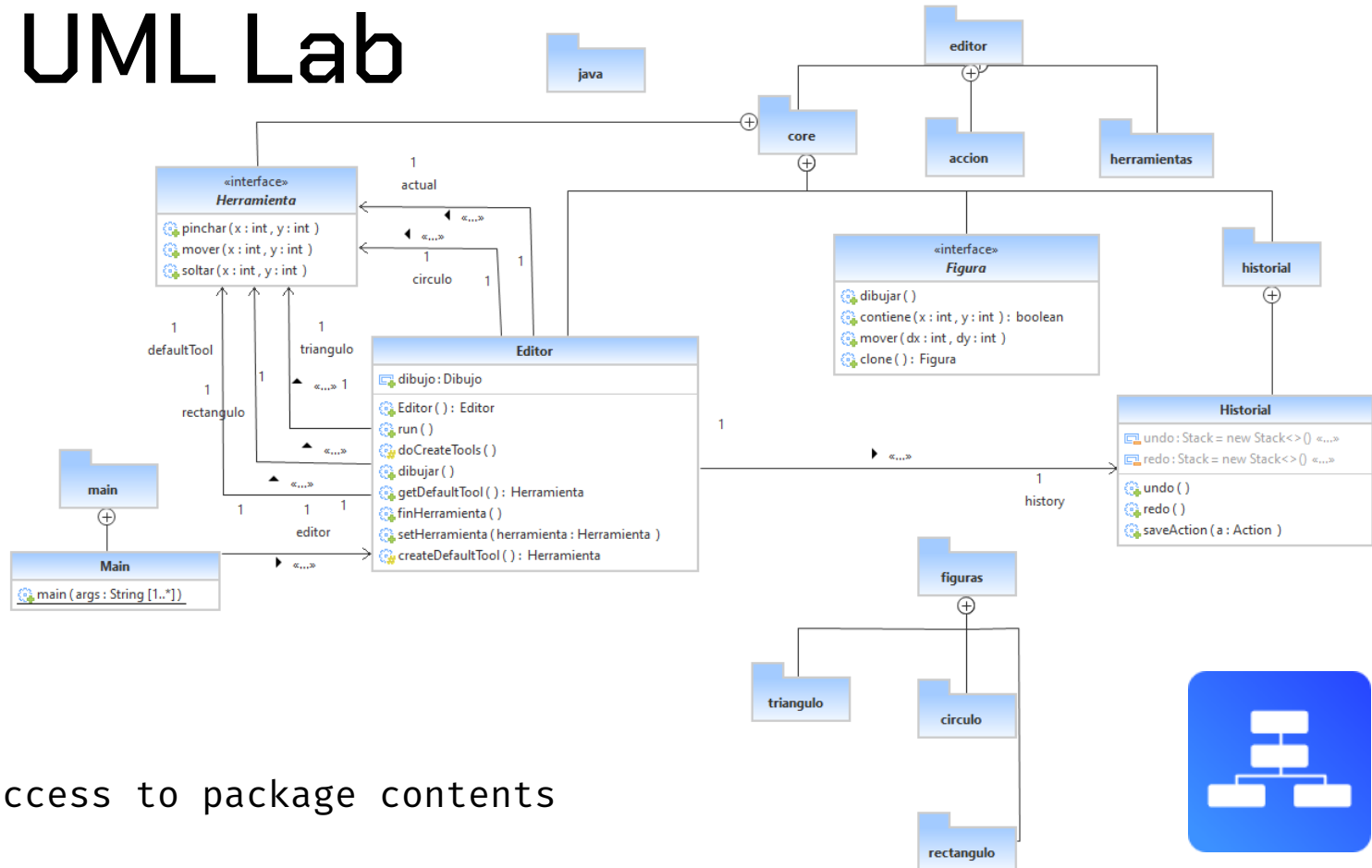


UML Lab

- Integrated with Eclipse IDE
- Diagrams synchronized with code



UML Lab



- **Dynamic** access to package contents

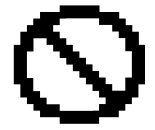




Any

question?





Thank you