

WP09

From Models to Views

Design of Software Architectures

Dr. Vadim Zaytsev aka @grammarware, 20 September 2022

Role of the software architecture



to provide
solution direction
for
most important properties
that are
the most difficult to realise



All models are wrong



View

- Representation of
 - a **system**
 - its **environment**
 - their **relation**
- from the perspective of
 - a set of related **concerns**
- **used** by an architect
 - to **define/explain/reason**



View = model? View ≠ model?

Model

- A *model* is
 - a *simplified* representation
 - of a *part* of the world
 - from a particular *view*
- Architecture model
 - source *specification*
 - for one or more *views*



Models

The screenshot shows a Jira Board for a project named 'SCIENT. TOOLS in S...'. The board is organized into columns representing different stages of the workflow: 'TO DO' (5 items), 'IN PROGRESS' (5 items), 'CODE REVIEW' (2 items), and 'DONE' (8 items). Each item includes a title, a status label (e.g., 'SPACE TRAVEL PARTNER'), and a due date. The interface includes navigation options like 'Roadmap', 'Backlog', and 'Active sprints'.

The screenshot displays the GitHub repository page for 'Nhaaj/negomancer'. It shows the repository's main page with options to view code, issues, pull requests, and actions. A list of recent commits is visible, including 'artifacts-ear Template cleanup', 'github Template cleanup', 'run Initial commit', 'grade/wrapper Initial commit', 'src/main Template cleanup', 'gitignore Initial commit', 'CHANGELOG.md Template cleanup', 'README.md Template cleanup', 'bulk.gradle.kts Initial commit', 'eetec-conf.yml Initial commit', 'grade.properties Initial commit', 'gradlew Initial commit', 'gradlew.bat Initial commit', and 'settings.gradle.kts Initial commit'. The repository also shows no published releases or packages.

```

1 module Example
2
3 imports
4
5 Common
6
7 context-free start-symbols
8
9 Start
10
11 context-free syntax
12
13 Start = Stmt
14 Stmt.If = <
15           if(<Expr>)
16             <Start>
17           else
18             <Start>
19
20 Expr.True = "true"
21 Expr.Var = ID
22 Expr.Gt = [[Expr] > [Expr]] {right}
23 Stmt.Assign = <<ID> = <Expr>
24 Expr.Int = INT
25 Expr.Minus = <<-<Expr>
26
27 context-free priorities
28 Expr.Minus > Expr.Gt
29
30 lexical syntax
31 ID = "true" {reject}
  
```

The code editor shows a grammar definition for a simple arithmetic language. It includes context-free start symbols, context-free syntax rules for statements and expressions, context-free priorities, and lexical syntax rules for identifiers. A tooltip for the 'if' rule shows its structure: 'if(Expr) { Expr }'.

```

1 import java.util.*;
2
3 class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.println("Enter a number:");
7         int n = scanner.nextInt();
8         System.out.println("You entered: " + n);
9     }
10 }
  
```

The screenshot shows a code editor with a Java class file named 'Main.java'. The code defines a simple application that prompts the user to enter a number and then prints the entered value. The IDE interface includes a project explorer on the left and a console at the bottom.

The screenshot shows the drawSQL software interface. It displays a database diagram for 'Keel - MySQL'. The diagram includes tables such as 'playlist', 'album', 'artist', and 'password_resets', with their respective columns and data types. The interface also shows a sidebar with table lists and a main workspace for editing the diagram.

The screenshot shows a Trello board with several lists and cards. The lists include 'Need More Effort', 'Writing', 'To Do', 'Opportunities', and 'Maybe'. Each list contains cards with titles, due dates, and other details. The board interface includes navigation options and a search bar.

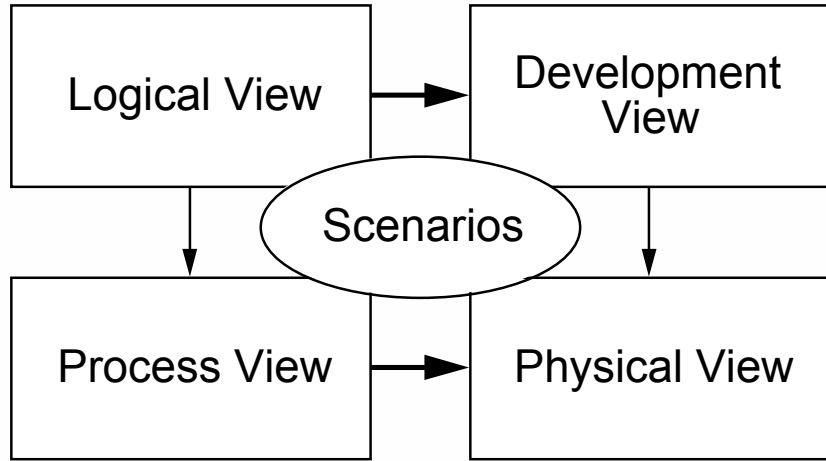
Viewpoint

- Way of looking at
 - system
 - environment
- from the perspective of
 - stakeholder
 - concern
 - aspect



End-user
Functionality

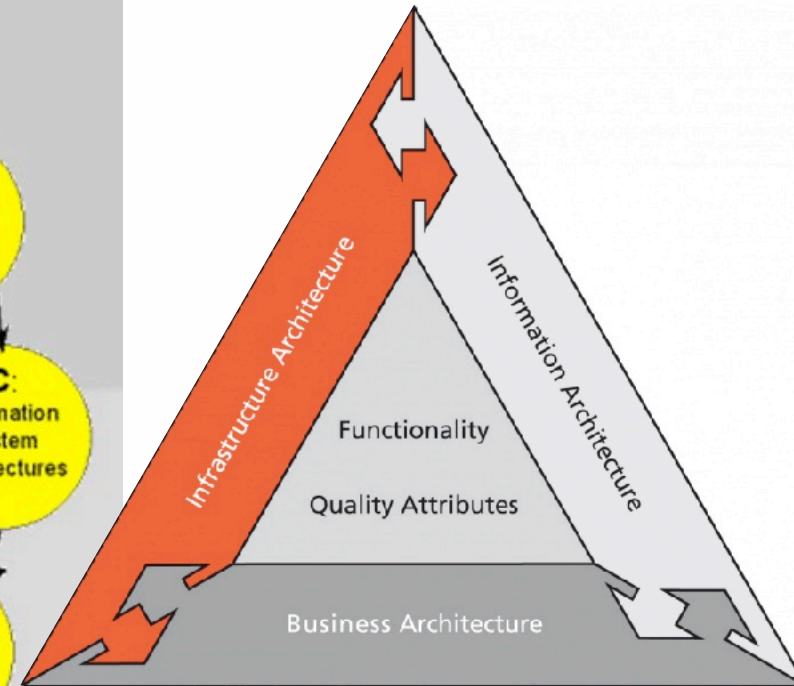
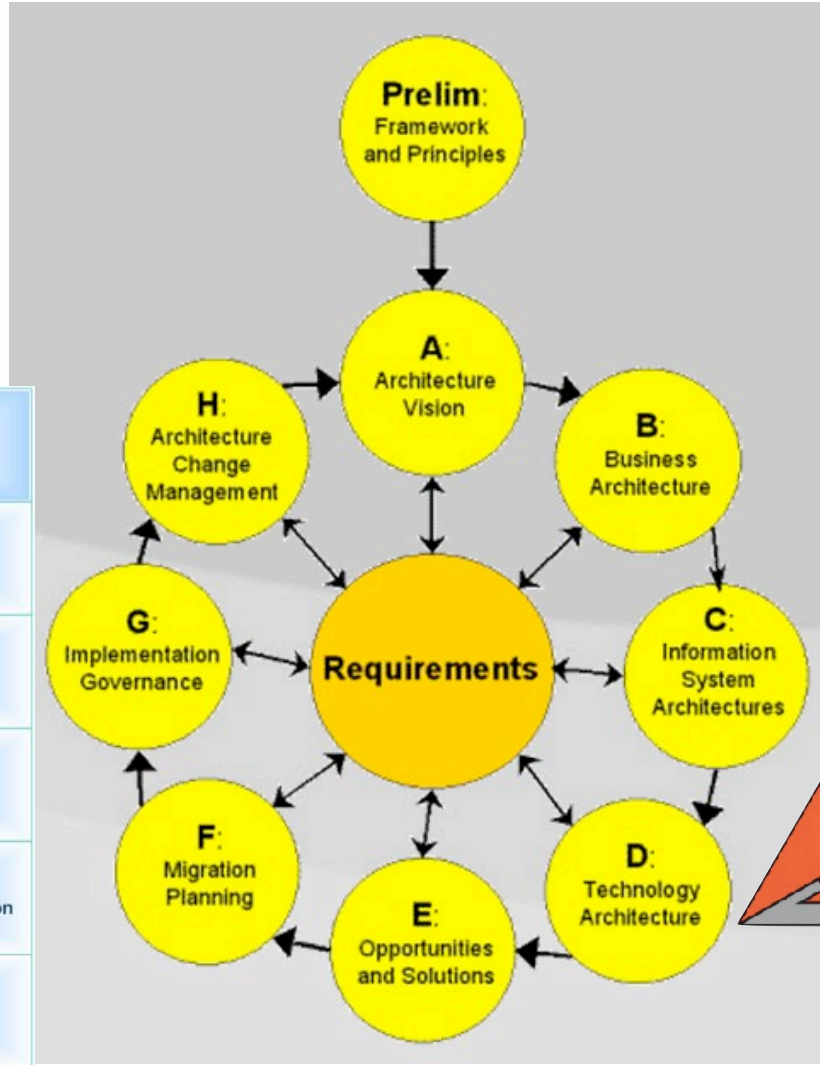
Programmers
Software management



Integrators
Performance
Scalability

System engineers
Topology
Communications

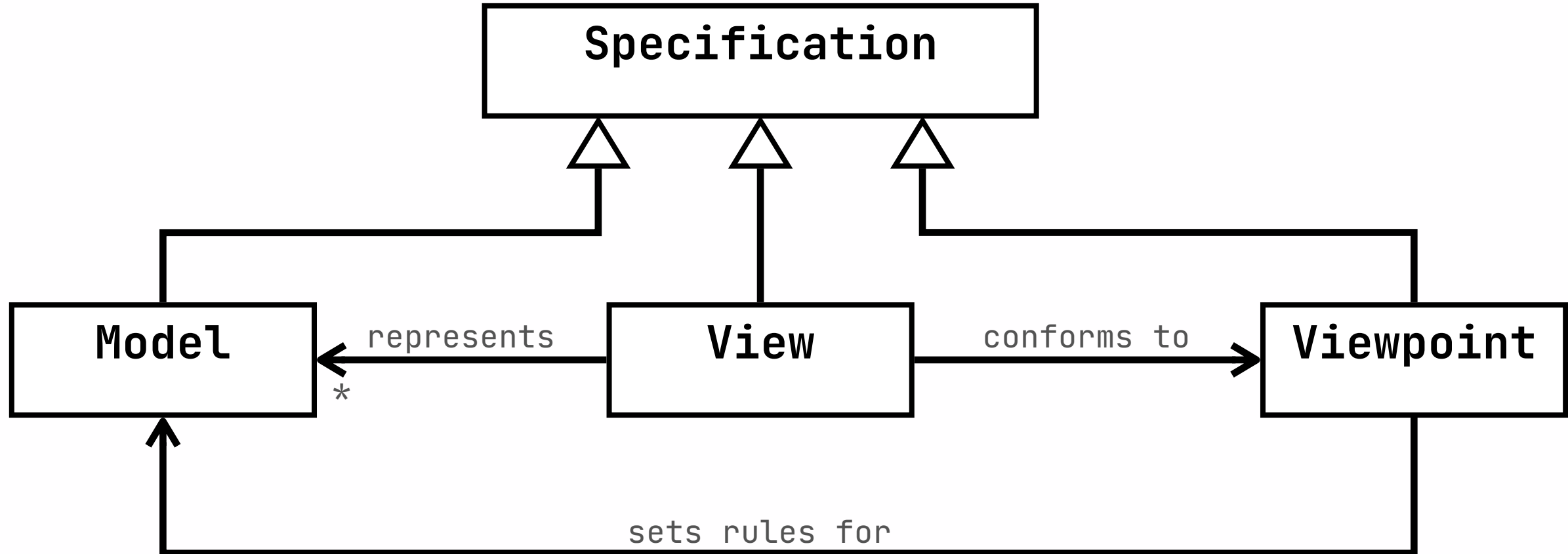
	Why	How	What	Who	Where	When
Contextual	Goal List	Process List	Material List	Organisational Unit & Role List	Geographical Locations List	Event List
Conceptual	Goal Relationship	Process Model	Entity Relationship Model	Organisational Unit & Role Relationship Model	Locations Model	Event Model
Logical	Rules Diagram	Process Diagram	Data Model Diagram	Role Relationship Diagram	Locations Diagram	Event Diagram
Physical	Rules Specification	Process Function Specification	Data Entity Specification	Role Specification	Location Specification	Event Specification
Detailed	Rules Details	Process Details	Data Details	Role Details	Location Details	Event Details



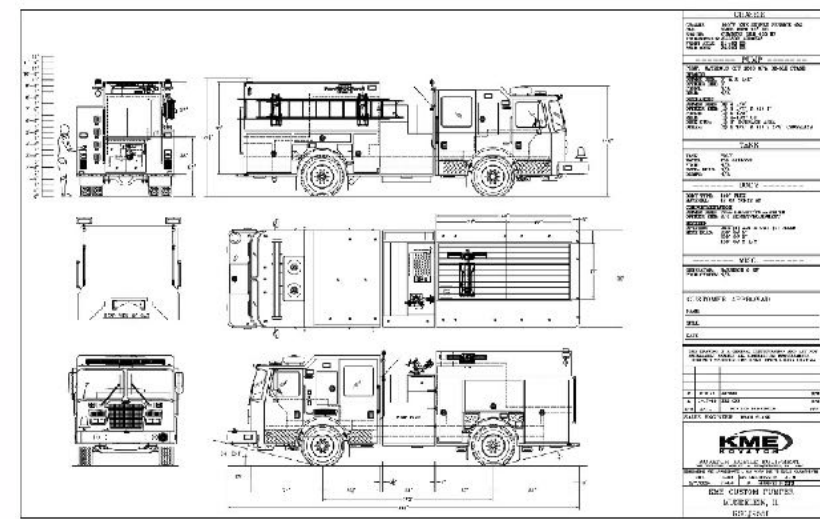
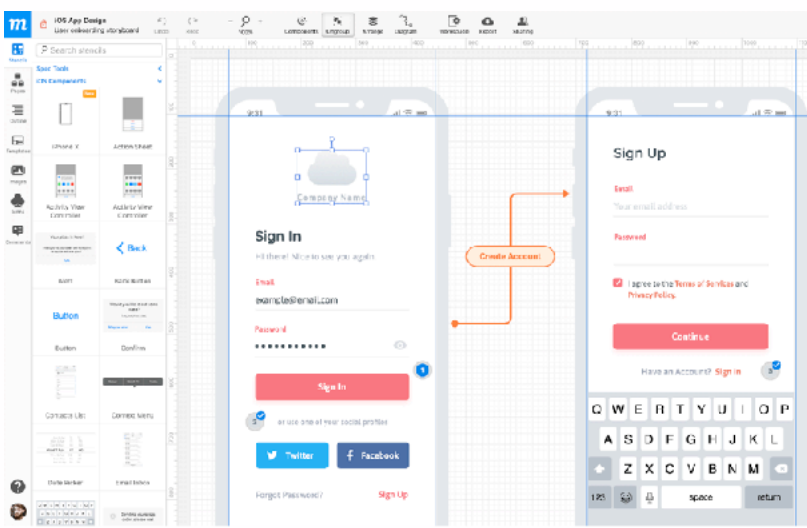
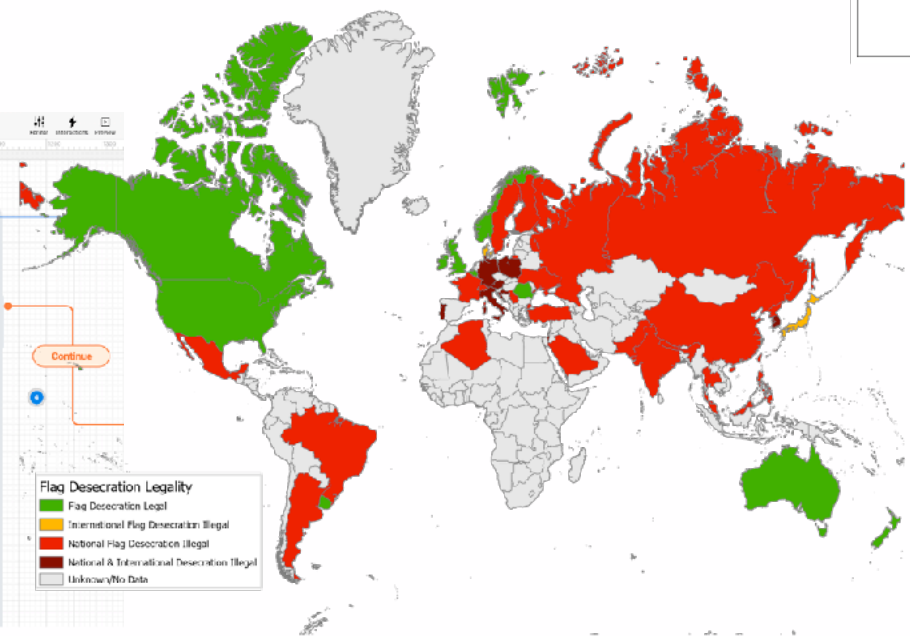
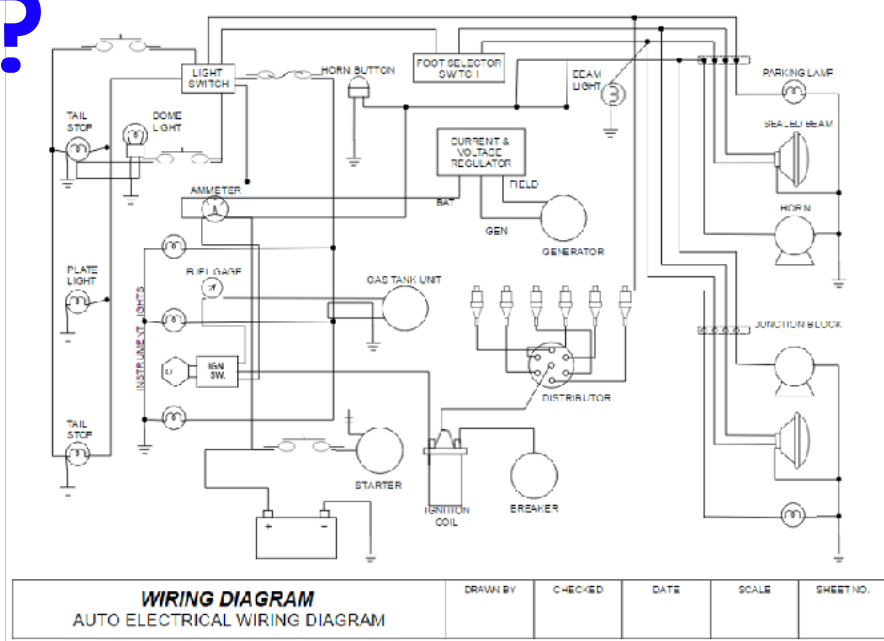
Viewpoints from a framework

- 4+1 model...
- TOGAF...
- DYA...
- Zachman...
- Standard framework does not cover **unique** properties
 - Use **specific** viewpoints
- Views can be related via the **architecture domain model**.

Model ← View → Viewpoint



Model? View? Viewpoint?



Select viewpoints, clean up views

- Which **stakeholder(s)**? Which **concerns**?
- Decide on the language:
 - **Concepts** and **notation**
 - **Visualisation** (depends on communication context)
- Specify how to translate the **model(s)** into a **view**.
- Pitfalls:
 - Too little views (e.g., just the decomposition)
 - Overloaded views (everything in one picture)

Remember!

- *An architecture description shall include each architecture viewpoint used therein.*
- *An architecture description shall include exactly one architecture view for each architecture viewpoint used.*
- *Each concern shall be framed by at least one viewpoint.*