APIs: semi-permeable, osmotic interfaces

Dawn Ahukanna, Design Principal and Front-end Architect

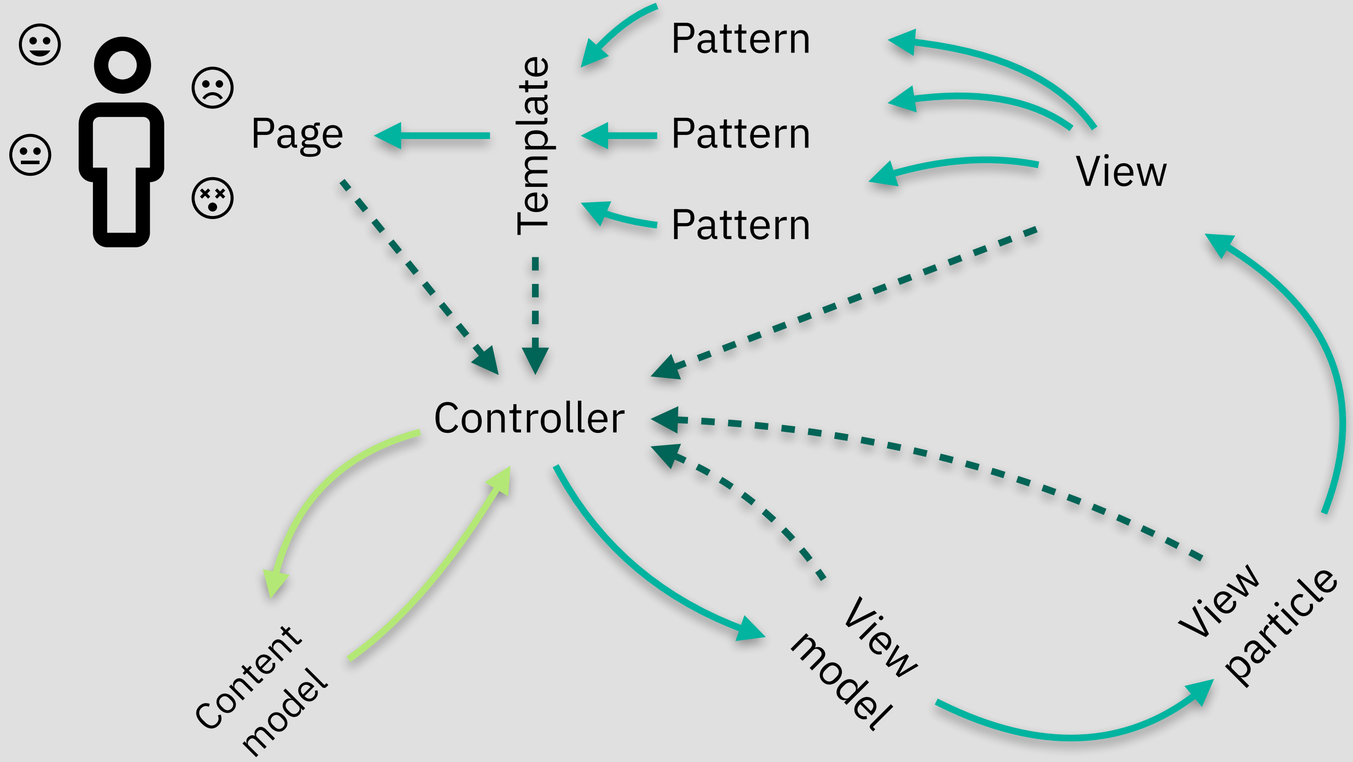
APIs: **Semi-permeable, osmotic interfaces**

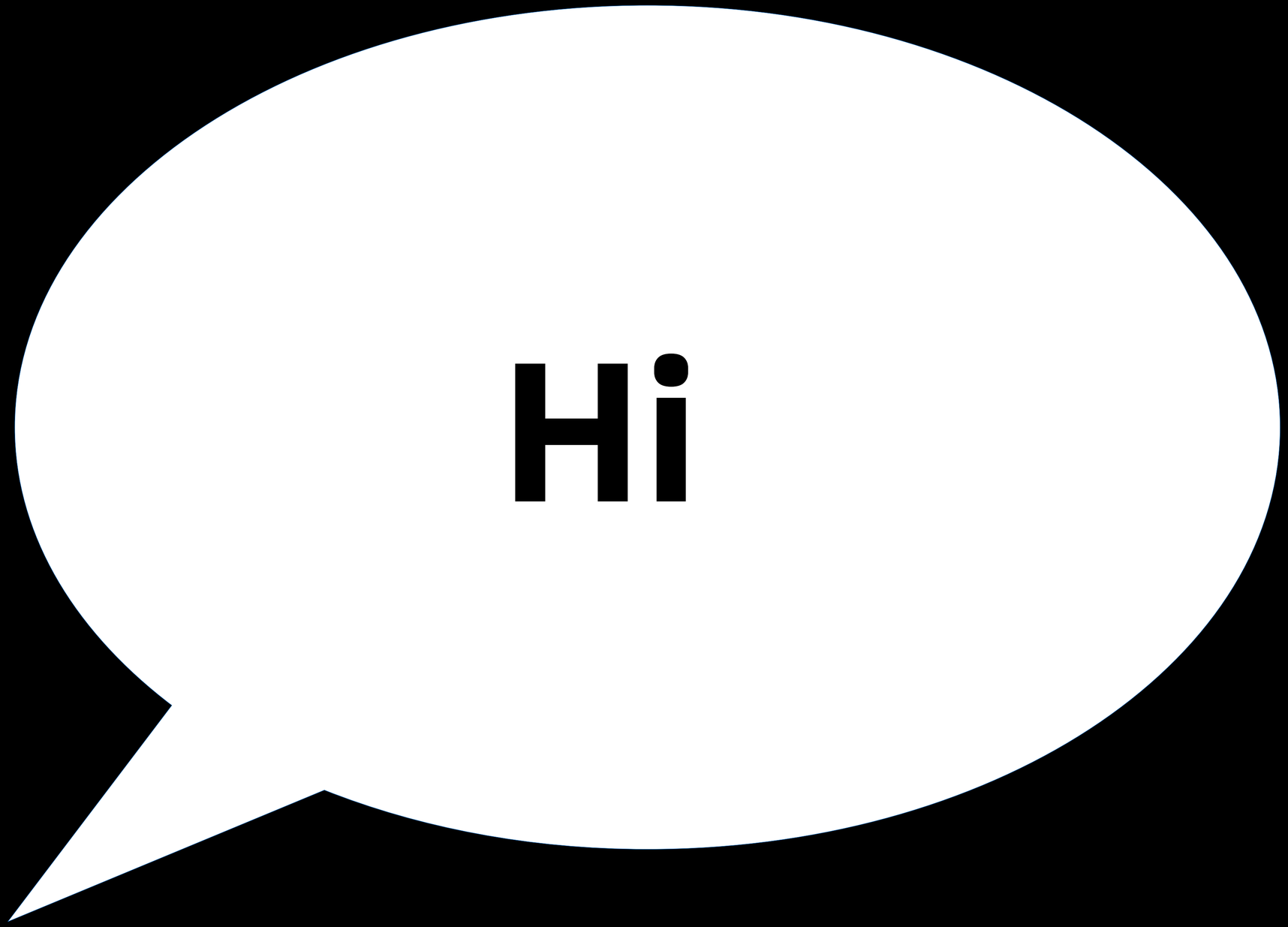
APIs can be modelled using the metaphor of semipermeable interaction boundaries between the intended High-Fidelity human-centered interfaces and the interpretation into a coded implementation.

Moving beyond “red-lining” static images, will discuss approaches and recommendations to consider the human developers and operators experiences with APIs, with membrane layers.

Agenda

- Introduction
- Definitions
- Membrane metaphor for API space
- Empirical example
  - Interface membrane layers
  - System membrane layers
- Q&A




Hi

Dawn Ahukanna

Design Principal and Front-End Architect

dawnahukanna

dahukanna

www.ibm.com/design/community/Dawn/

dawnahukanna

I’m a Computational Designer.
Computational Designers are creative people **applying** their **mastery** and **knowledge** of how people, software, hardware, data, network and systems all interact as **one closed system**, with an operating, open system environment.

Working as a Design Principal in IBM. Helping clients in partnership with business partners to simplify IT management and operations using preventive maintenance, analytics and Deep Learning probabilistic insights.

Definitions

Application Programming Interface · API

(Definition):

Application programming interface (API) is a mechanism for two or more computer software programs to connect, communicate, transport, transfer and transform transported objects, to create a function capable of doing some work.

——

APIs are usually specified individually, without suggested sequencing and dependencies.

Semipermeable membrane

(Definition):

Semipermeable membrane is a biological or synthetic layer that allows only certain molecules (objects) to pass through.

Osmosis

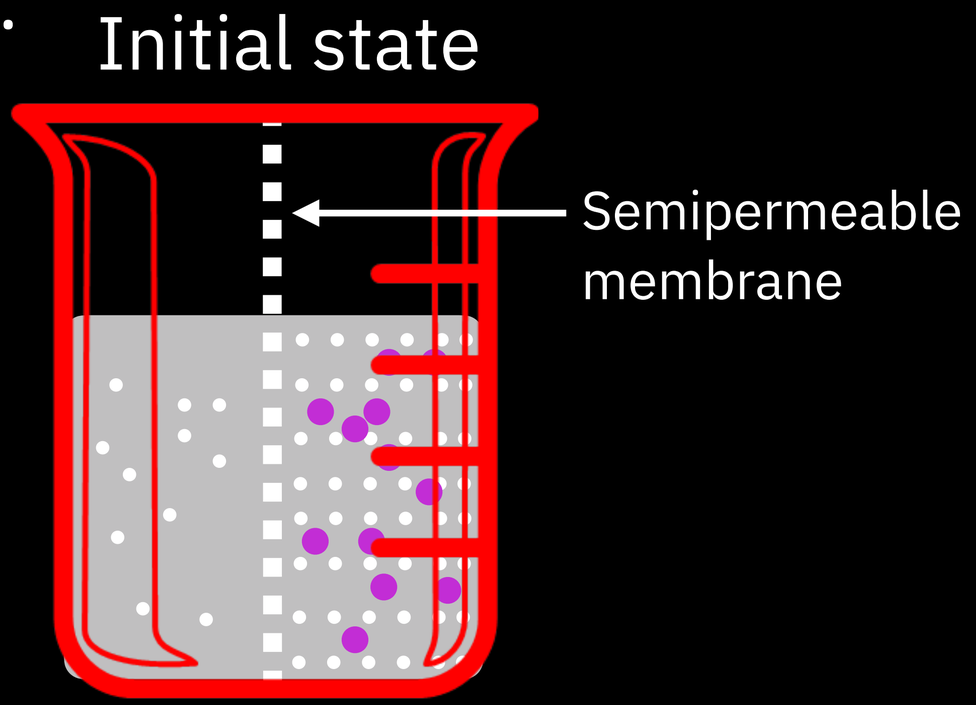
(Definition):

Osmosis is a process where molecules of a liquid pass thorough a semipermeable membrane. From a less concentrated solution to a more concentrated one.

Osmosis

(Definition):

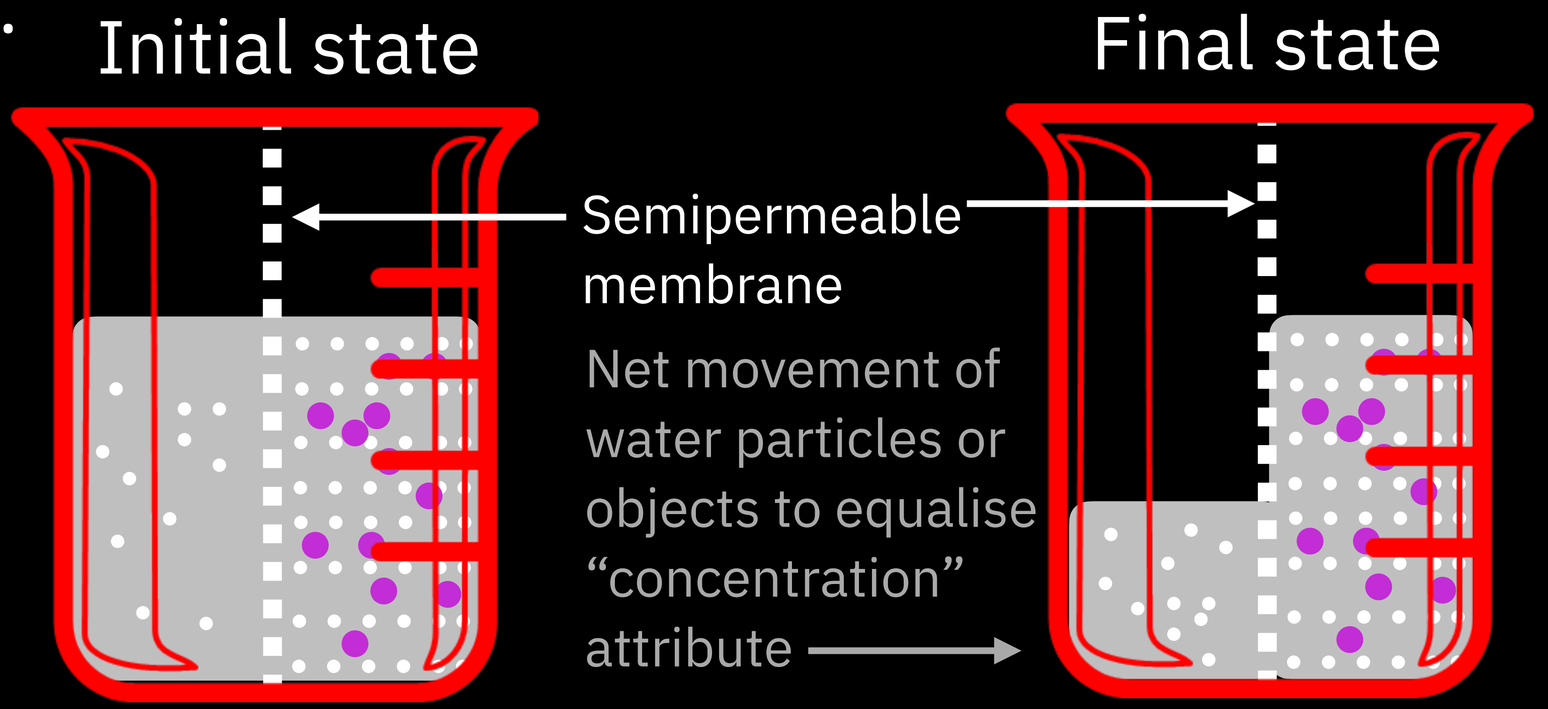
Osmosis is a process where molecules of a liquid pass thorough a semipermeable membrane. From a less concentrated solution to a more concentrated one.



Osmosis

(Definition):

Osmosis is a process where molecules of a liquid pass thorough a semipermeable membrane. From a less concentrated solution to a more concentrated one.



Cognitive Dissonance

(Definition):

Cognitive dissonance is a psychological term describing the mental conflict that occurs when a person holds two beliefs that contradict each other.

——

Some perspectives may seem to contradict each other, when they are really flattened planer orientations of the spectrum of possible values in a space e.g. hot is the opposite of cold. Something can be hot and cold at the same instant, depending on where you measure.

Assumption

(Definition):

Assumption is a statement or thing that is accepted as true or certain of occurring, without proof.

——

I’m on a mission and recommend using presumptions instead: a statement or thing that is accepted as true or certain of occurring, on the basis of probability. The nature of probability is that is takes into account all variables active at the point in time the probability of occurrence was assessed.

Membrane Metaphors for API space

3D-cube

Think and model each challenge that your solving as a flattened 3-D space. Each flattened section is a semi-permeable membrane that only allows certain “molecule” objects to pass through.

“The things you tend to **dismiss as mistakes** perhaps you **should take more seriously**.

They **weren’t** what you **intended**, but that does **not** make them **wrong**.”

– Brain Eno,
Musician and Music Producer

“The things you tend to **dismiss as mistakes** perhaps you **should take more seriously**.

They **weren’t** what you **intended**, but that does **not** make them **wrong**.”
- or **complexity,** maybe “it’s **complicated**”?

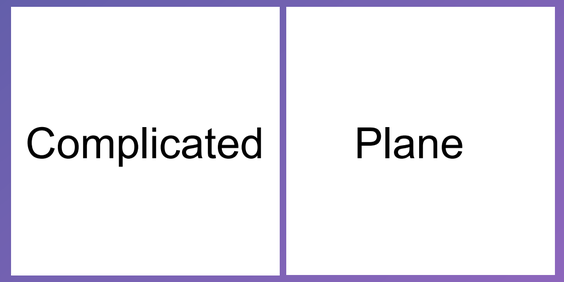
– Brain Eno,
Musician and Music Producer

Complicated membrane

Consisting of many interconnected parts or elements.

Also, complicated is past tense. Therefore regardless of number and intricacies of the parts, all factors are known. The known unknowns are accounted for by stated and validated assumptions.

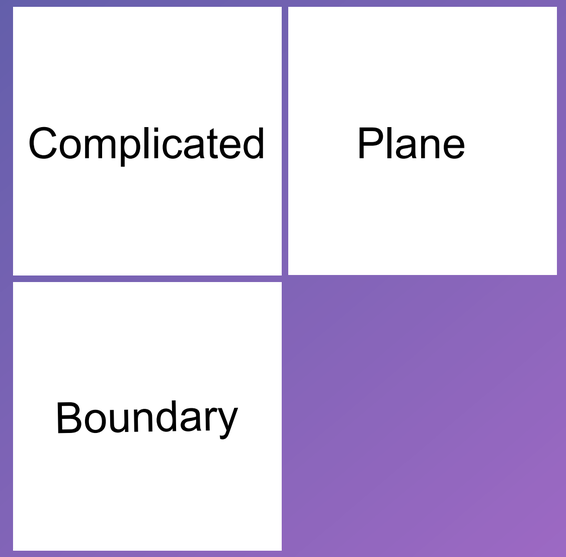
Plane or planer area membrane



From mathematics, a plane is a flat, 2 dimensional surface that can extend to infinity.

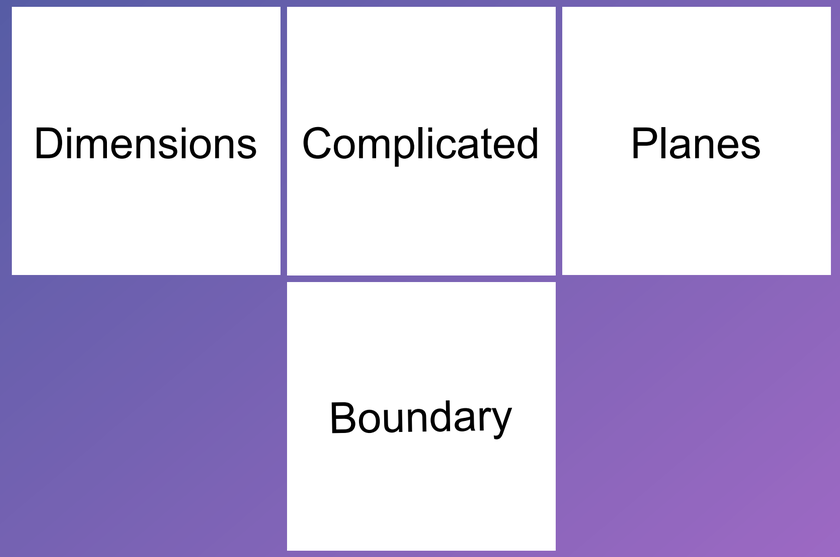
Boundary membrane

From mathematics, the limits of a planer area or a dividing line.



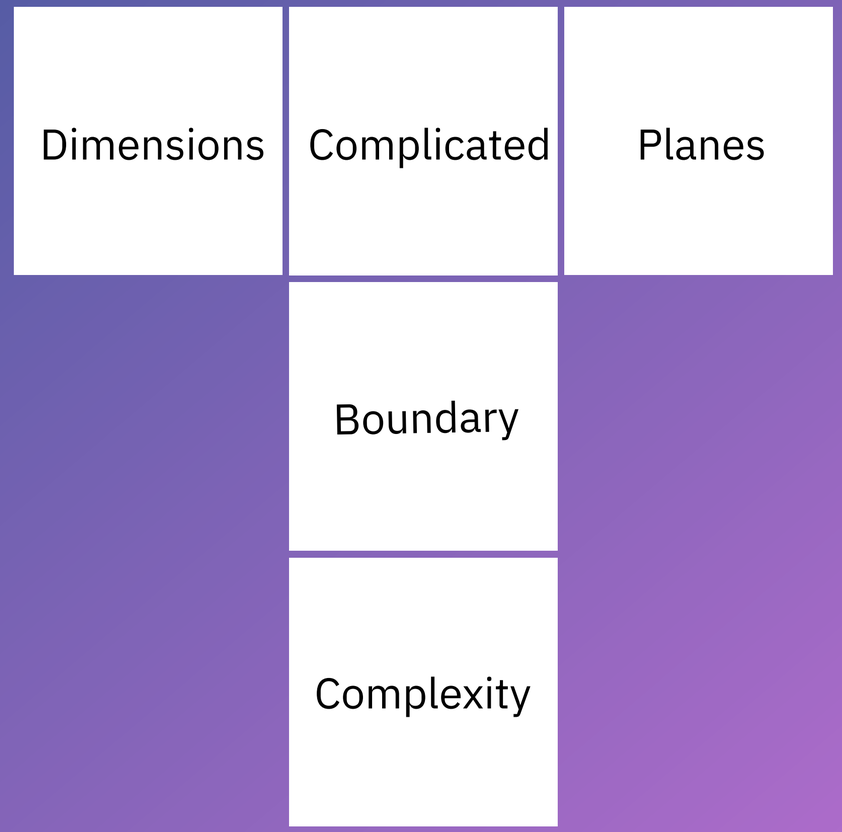
Dimensions membrane

From mathematics or physics, the the minimum number of coordinates required to specify a point within that space.



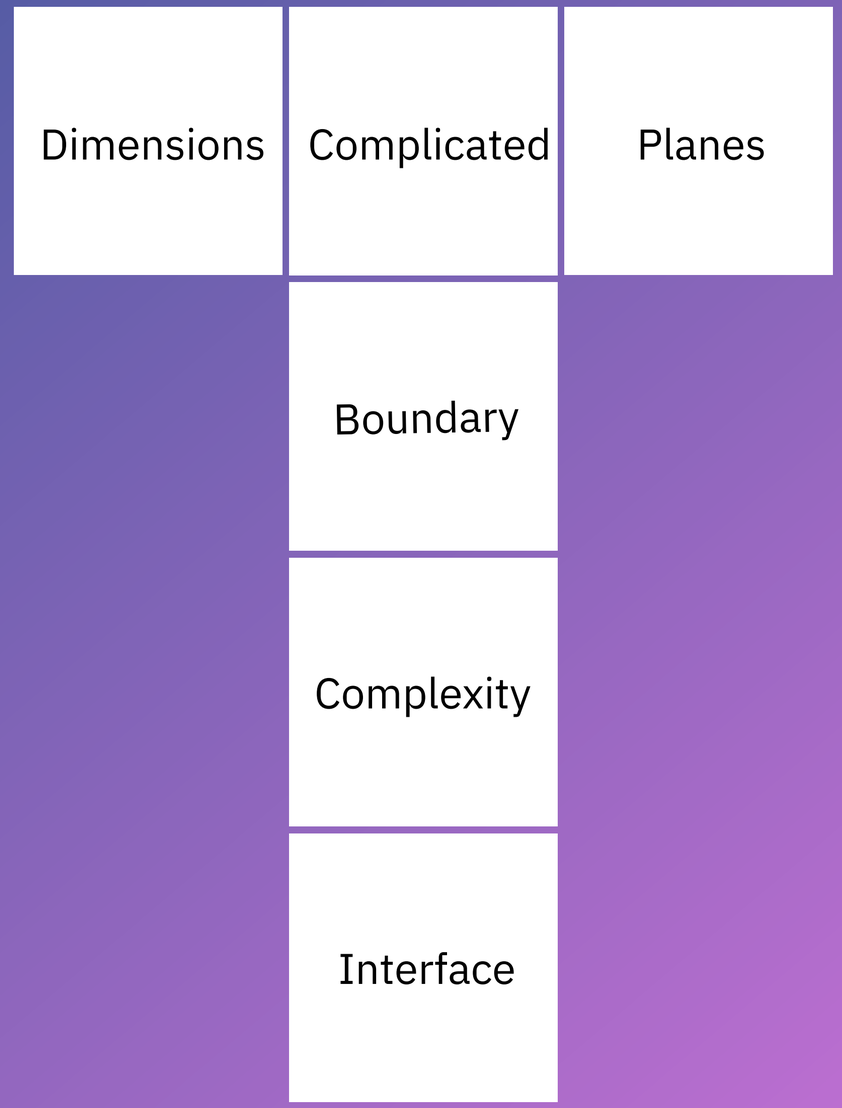
Complexity membrane

From mathematics, involving numbers containing both a real (known) and an imaginary (unknown) part.



Interface membrane

A point where 2 or more system, organisations, people meet and interact.

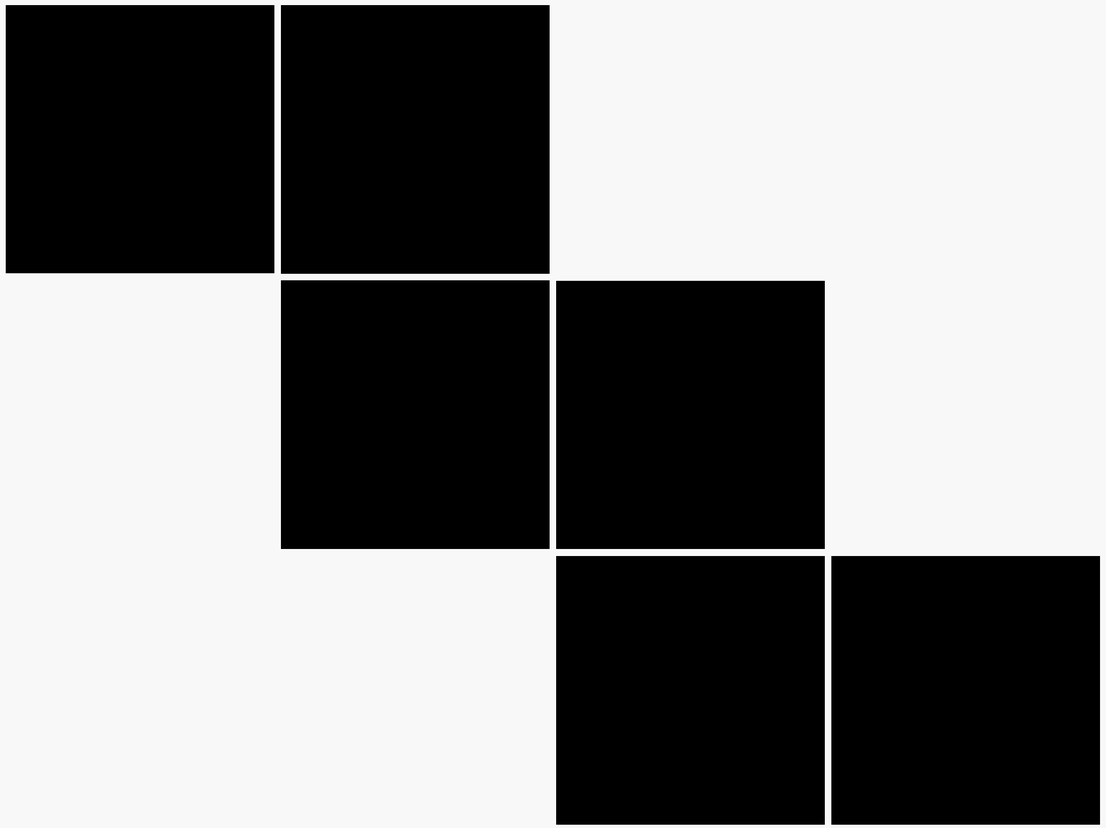


Flattening or abstracting to simplify

The same set of square units than compose a cubed space could be unfolded in a different, valid number of ways that results in the same 3-D shape but in dissimilar looking 2-D shapes.

Expressed mathematically:
Area = length x breadth and Volume = length x breadth x height are the same value.
But the planer arrangement is a different orientation.

Which arrangement of interface or membrane category distinction is important for your purpose or context?

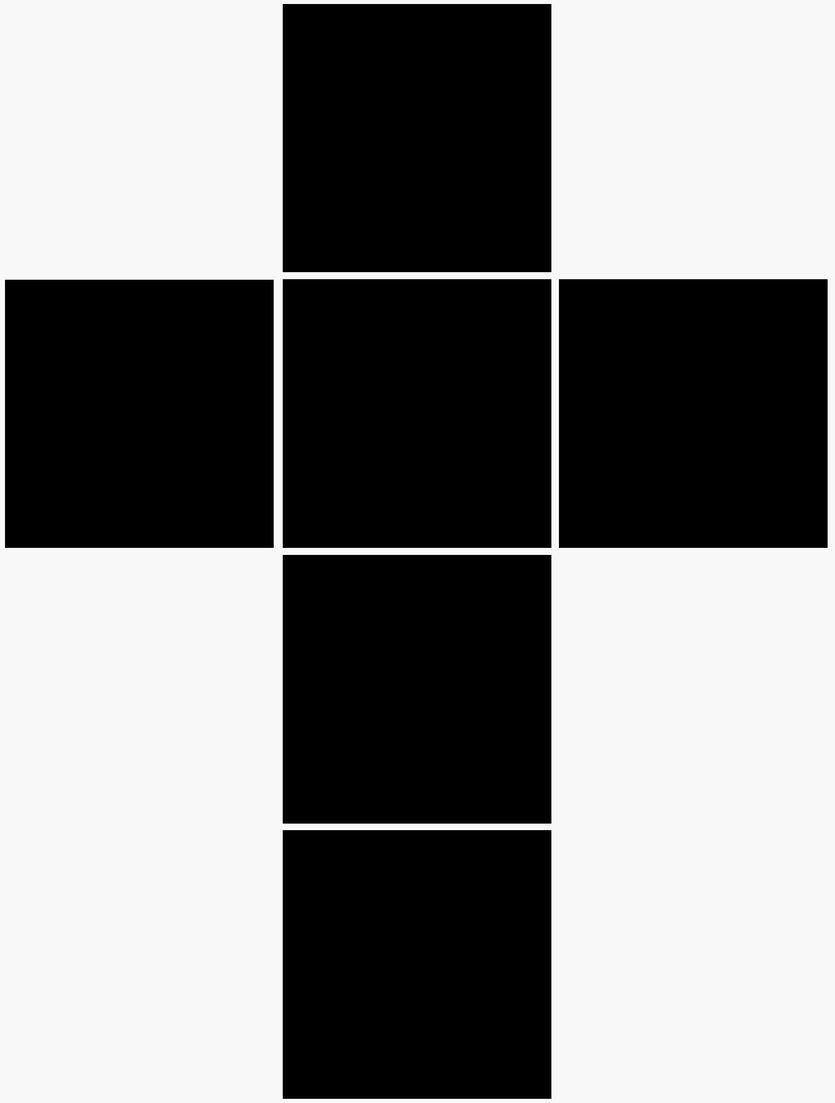


Flattening or abstracting to simplify

The same set of square units than compose a cubed space could be unfolded in a different, valid number of ways that results in the same 3-D shape but in dissimilar looking 2-D shapes.

Expressed mathematically:
Area = length x breadth and Volume = length x breadth x height are the same value.
But the planer arrangement is a different orientation.

Which arrangement of interface or membrane category distinction is important for your purpose or context?

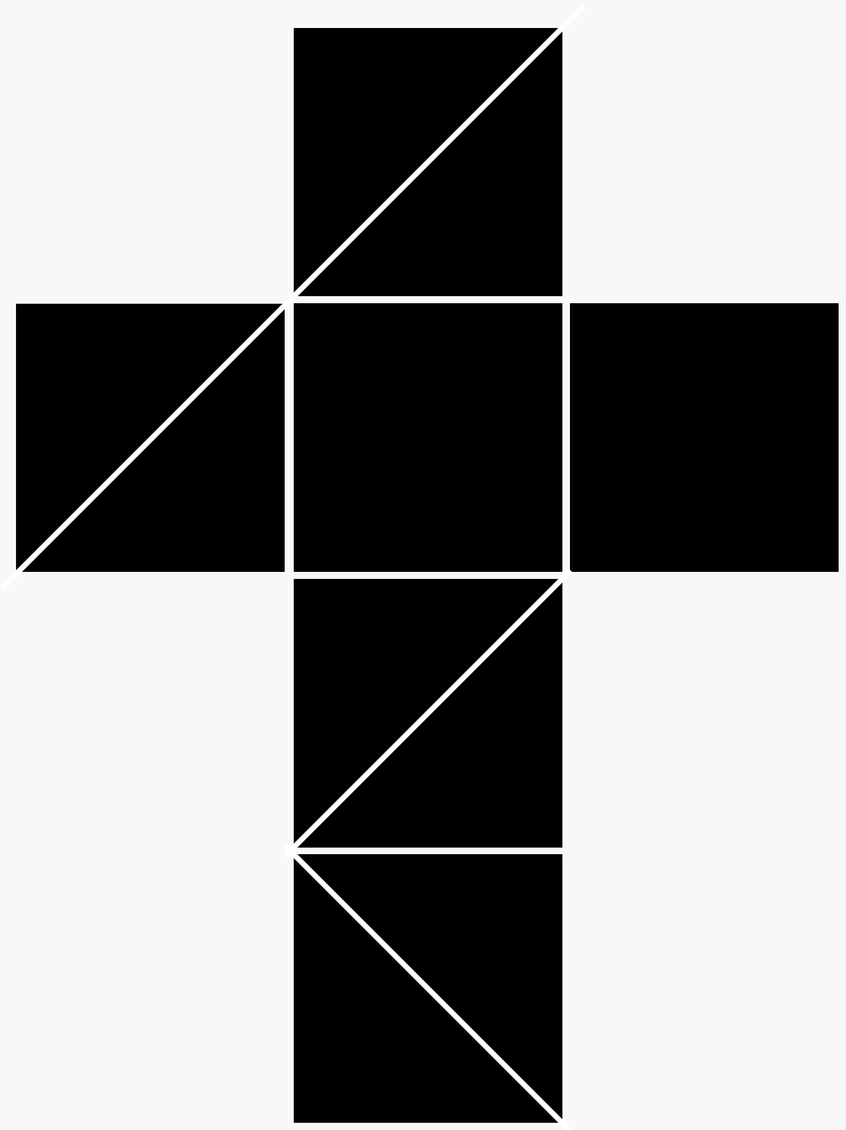


Flattening or abstracting to simplify

The same set of square units than compose a cubed space could be unfolded in a different, valid number of ways that results in the same 3-D shape but in dissimilar looking 2-D shapes.

Expressed mathematically:
Area = length x breadth and Volume = length x breadth x height are the same value.
But the planer arrangement is a different orientation.

Which arrangement of interface or membrane category distinction is important for your purpose or context?

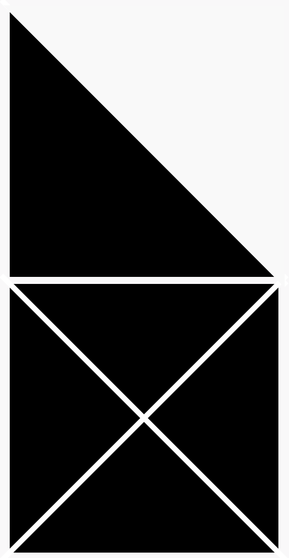


Flattening or abstracting to simplify

The same set of square units than compose a cubed space could be unfolded in a different, valid number of ways that results in the same 3-D shape but in dissimilar looking 2-D shapes.

Expressed mathematically:
Area = length x breadth and Volume = length x breadth x height are the same value.
But the planer arrangement is a different orientation.

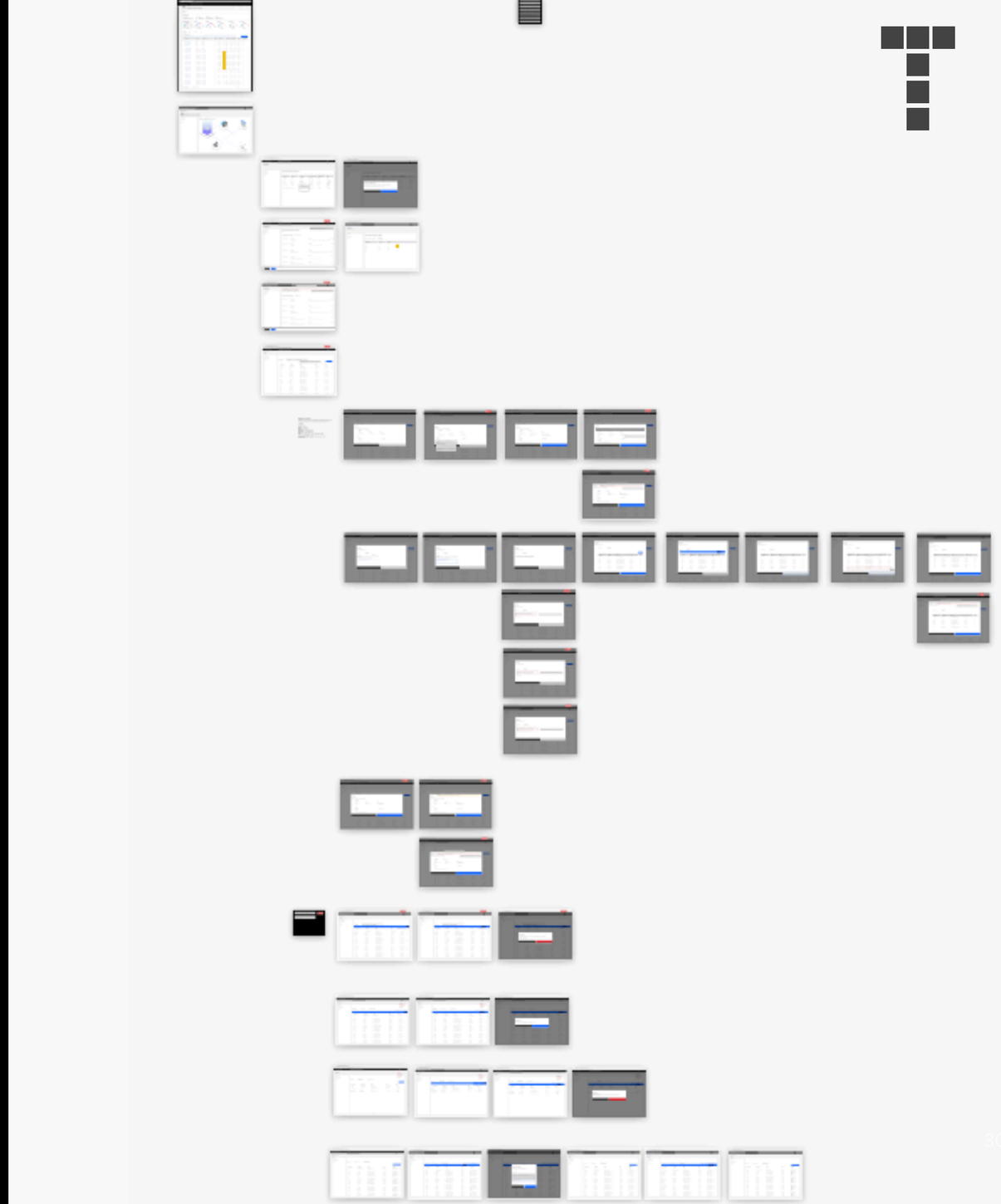
Which arrangement of interface or membrane category distinction is important for your purpose or context?



Empirical example

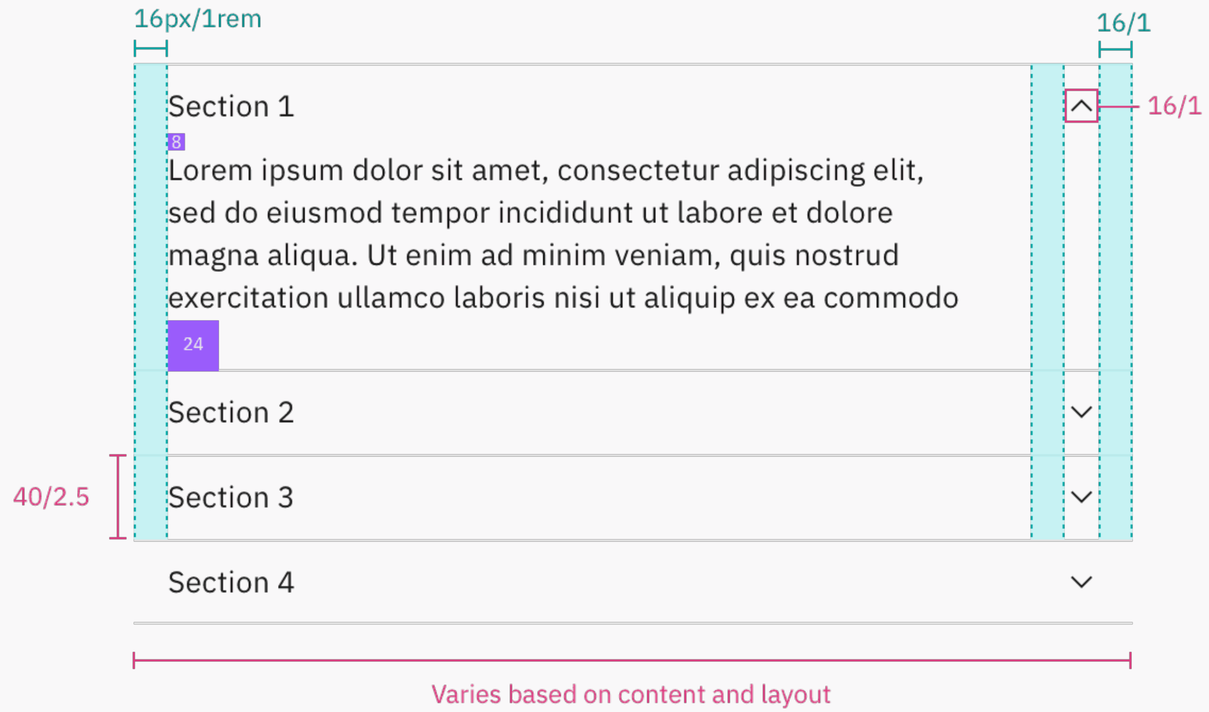
Planer interfaces

Would usually use the planer or flattening practice of “red-lining” static screen images to indicate interactions, connections, transfer and dimensions within the interface membrane and other system membranes (APIs).



Planer interfaces

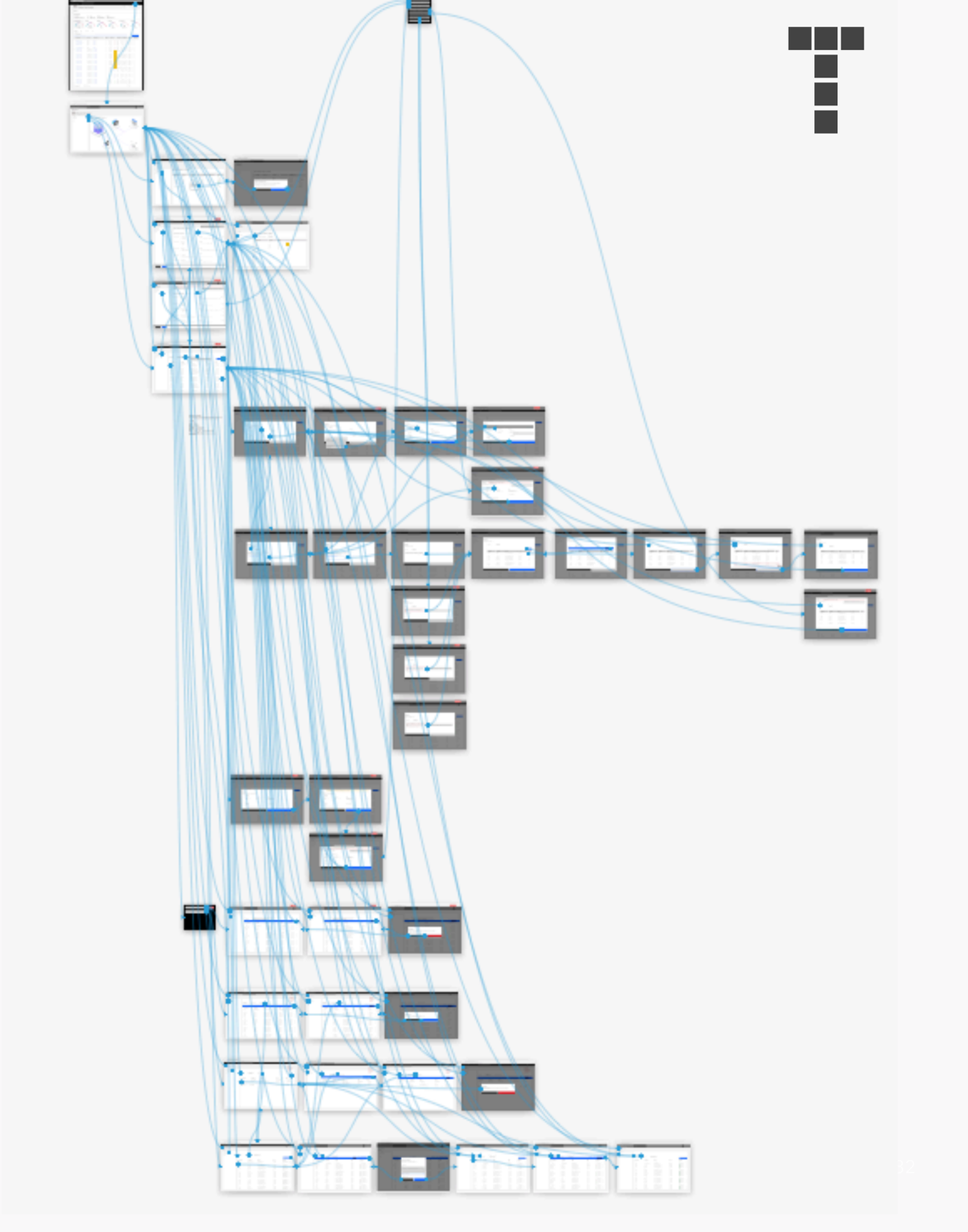
Would usually use the planer or flattening practice of “red-lining” static screen images to indicate interactions, connections, transfer and dimensions within the interface membrane and other system membranes (APIs).



Planer interface interactions

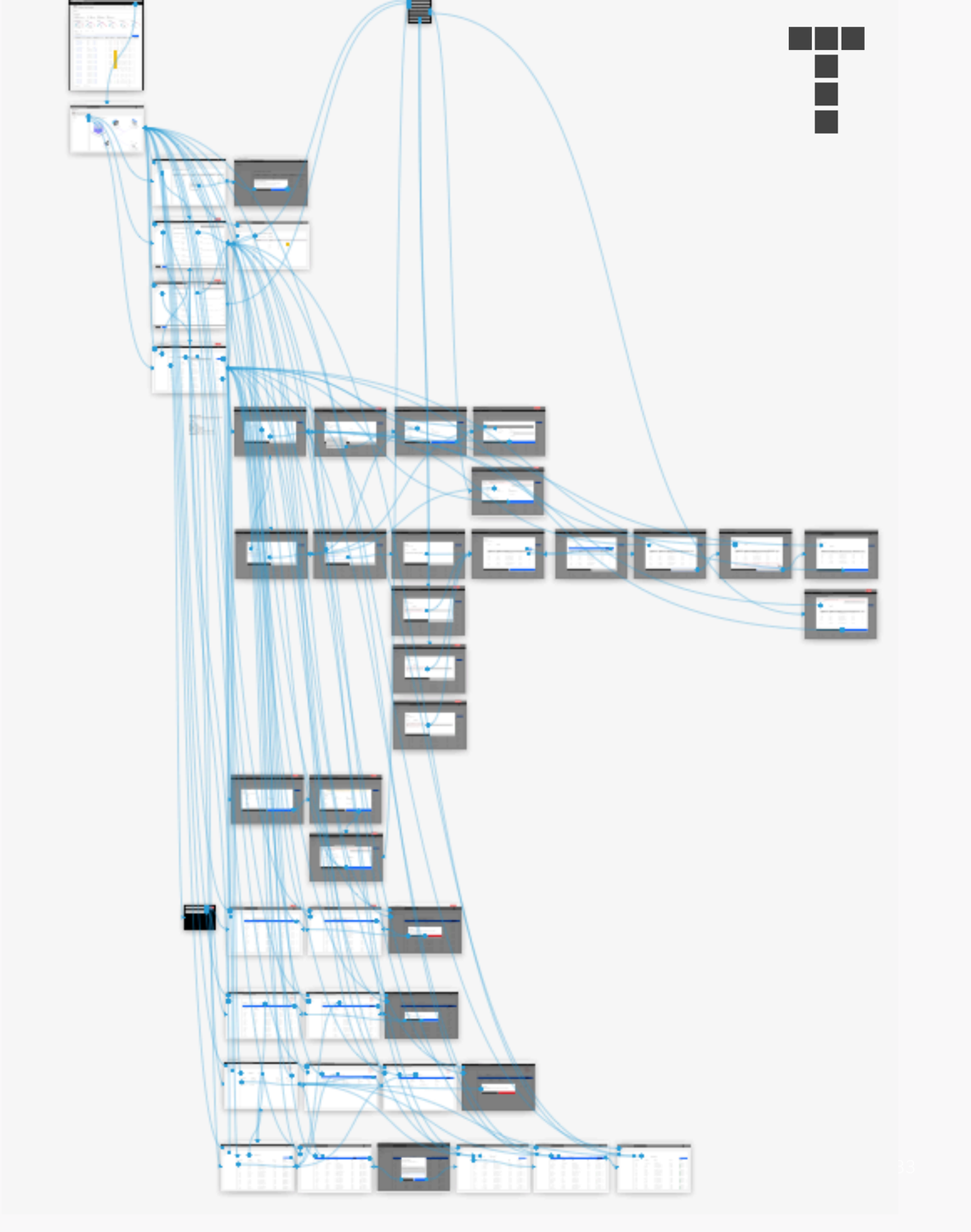
Each line defines a one dimensional pathway of interaction between two screen planes where everything occurs perfectly and there are no exceptions.

Design “spaghetti” interaction interfaces.



Planer interface interactions

One of the additional challenges is threading the system level interaction membranes (APIs) that are not documented in this diagram, to indicate API call sequencing and dependencies e.g. when and how to use security token generation, usage and expiration.



Interface Membrane Layers

Atomic Design System

invented by Brad Frost

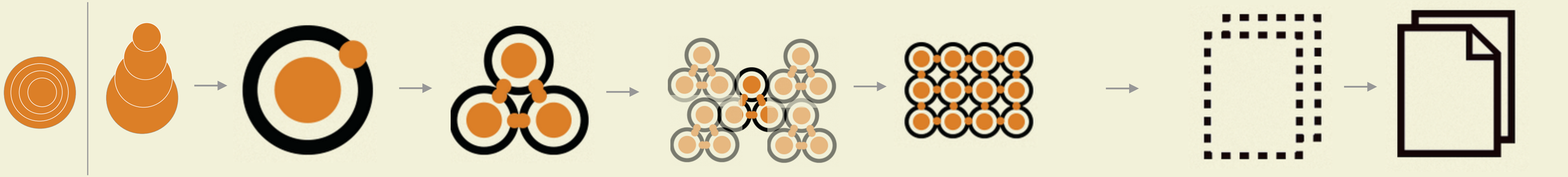
modified by me

Composable layers: containing the composites for particles, atoms, molecules, compounds and organisms

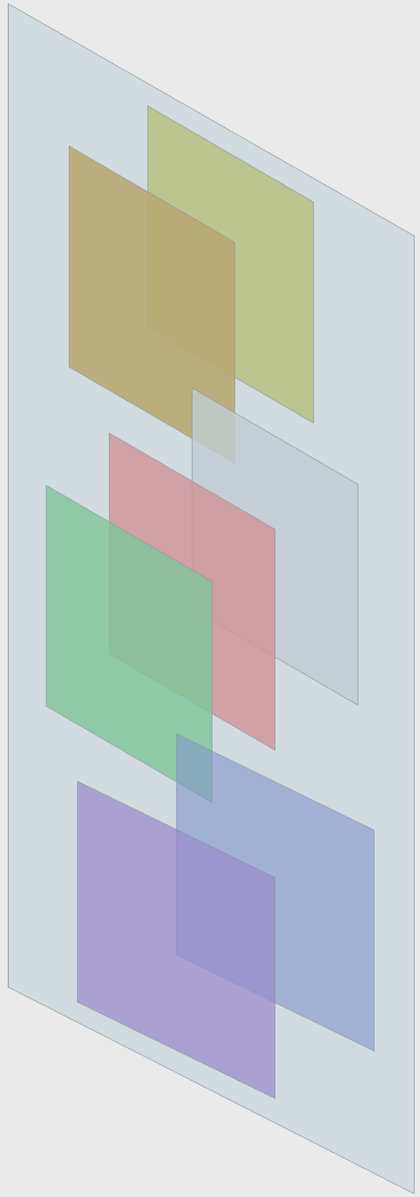
Membrane:

Composition: Covered by defined Design guidelines, components, UX and workflow patterns

Placement and flow

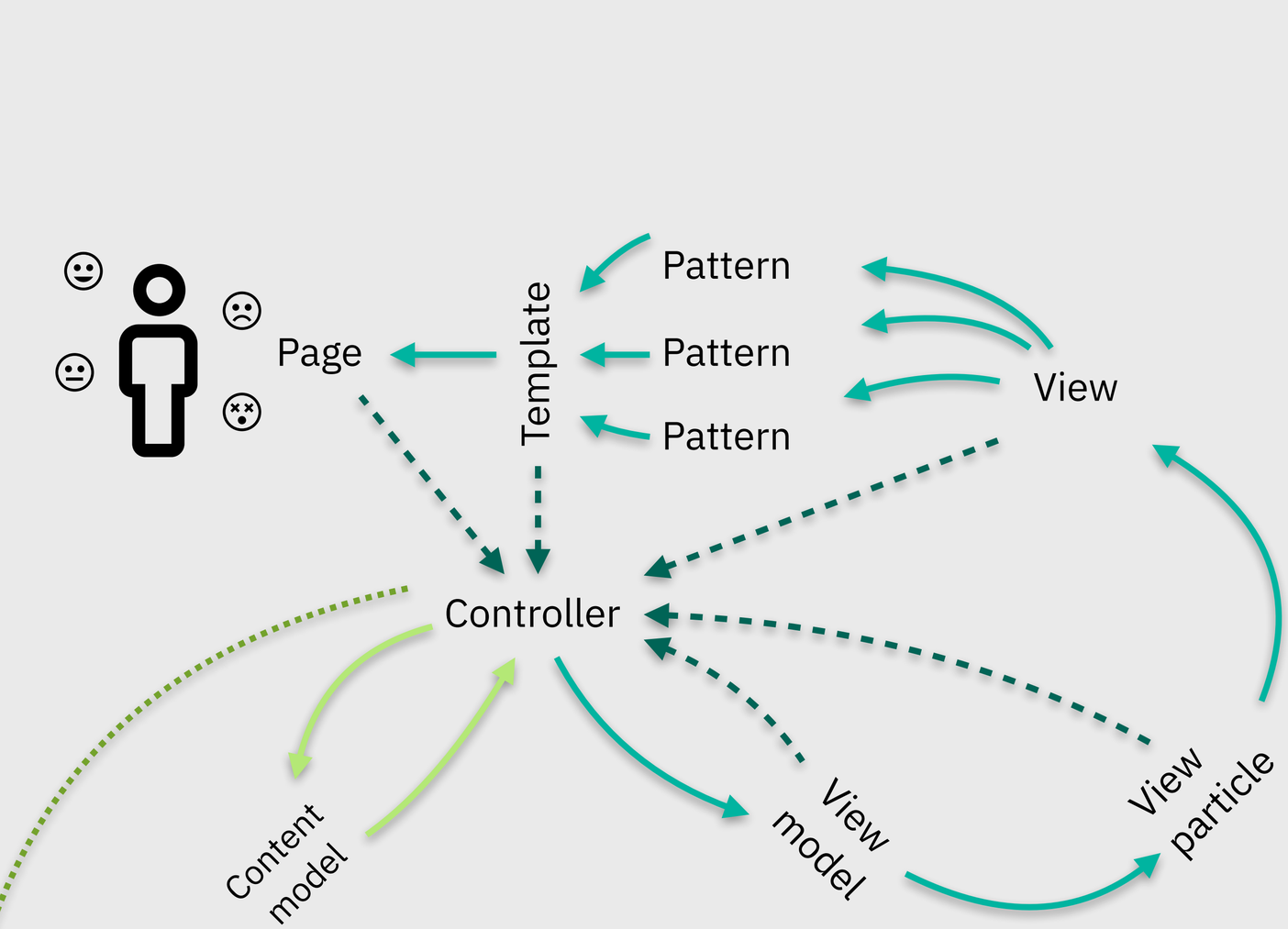


Particles or Elements **(added)**	Atoms	Molecules	Compound **(added)**	Organism	Templates	Pages or Screens
Colours Fonts & styling Backgrounds Borders ...	Buttons Dropdown ...	Combined atoms e.g. Header avatar == Layout + Image for avatar + Label for welcome + Text for avatar name	Combined molecules	Combined compounds	Layouts includiing grids for organisms, compounds, molecules, atoms and particles with placeholder content.	Specific instances of templates, with representative content replacing placeholder text



Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Pattern
- Template
- Page



External system integration API

Data integration API

App Interaction API

App Communication API

Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Pattern
- Template
- Page

Scenario

Going through the interface layers, starting farthest away from the final destination, the screen.

Content layer (Data and State)

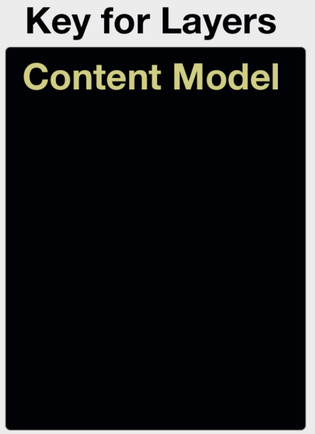
Data supplied or sourced from an API/Database/ service, etc. for evaluating, processing, transforming and eventually displaying on the page via a component.

Could leverage services for system and/or user security, data integration, data caching, data storage and interaction state management.

For example:

1. Content supplied from a data connection, communication, transport, transfer and transform API request.


```
{
  "updatedTime": 1516788000000,
  "data": {
    "next": [16,48,16,0,0,36,52,68],
    "previous": [16,156,94,0,0,12,12,12],
    "time": [1516834800000,1516921200000,1517007600000,1517094000000,1517180400000,1517266800000,1517353200000,1517392800000]
  }
}
```

Content Controller layer (Logic)

Logic, functionality, artificial intelligence and so on that processes, transforms and stores the Content Model source data before supplying it to a view.

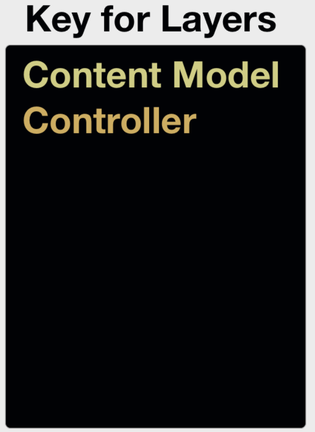
For example:

1. Format the data.
2. Derive values from the raw data.
3. Transform numbers to strings

e.g. Transform and format **1516788000000 to January 23 2018.**

```
Source: 1516834800000 (epoch time) validated and data formatted as English date MMM DD YYYY,
based on either default(static-fixed) or user formatting (dynamic-real time) preferences.

Transformed and Formatted Output:January 23 2018
```



View Model layer (State and Content)

The data and it’s interaction state are stored in a view model. This can be sent to one or more components for eventual display.

For example:

1. Store the transformed and formatted date label or date value in a view model.

Could leverage services for system and/or user security/access, data caching, data storage and interaction state management.

Key for Layers

Content Model
Controller
View Model

```
View model data value: January 23 2018
```

View Particle layer (Basic building blocks)

The styling, layout, interactions and state representing a display building block at the lowest level.

Key for Layers

- Content Model
- Controller
- View Model
- View Particle

View Particle layer (Basic building blocks)

The styling, layout, interactions and state representing a display building block at it's lowest level.

For example:

1. Retrieve the relevant view model, that has prepared the date for display within a component.
2. Format with specified typography, style the bounding box background and boundary. component, and apply view layout and styling overrides.
2. Apply icons, apply themes and other styling elements.

Particles are basic UI building blocks for things like typography, colours, etc.

Key for Layers

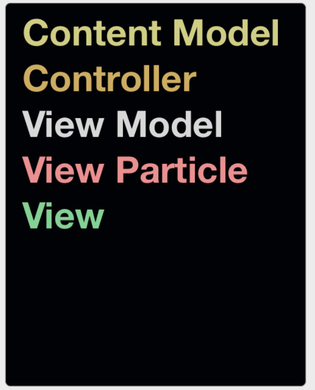
Content Model
Controller
View Model
View Particle

`View model data value: January 23 2018` + Controller Function: [Format date value with relevant and specified Typography, colours, etc] = "January 23 2018"

These are all **view particles:** "January 23 2018"

View Style and Interaction layer

Views (atomic components) and nested-views (molecular components) made up of view building blocks that includes content/data, define the behaviour of interaction and styling related to items that make up components.



View Style and Interaction layer

Views (atomic components) and nested-views (molecular components) made up of view building blocks that includes content/data, define the behaviour of interaction and styling related to items that make up components.

For example: Format, background, borders, behaviour and layout of -

1. Primary and secondary buttons
2. Checkboxes
3. Textboxes
4. Text Areas
5. Text Strings

Design System supplies a number of foundational components.

Key for Layers

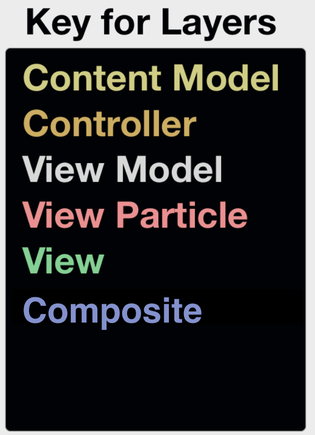
- Content Model
- Controller
- View Model
- View Particle
- View

Composite layer (Modular Components)

Atomic components in specific and validated combinations to create composable views or molecules.

For example:

Assemble configured and styled components to create reusable units.



Composite layer (Modular Components)

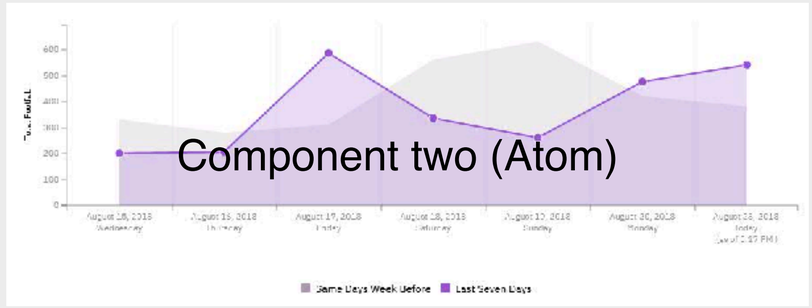
Atomic components in specific and validated combinations to create composable views or molecules.

For example:

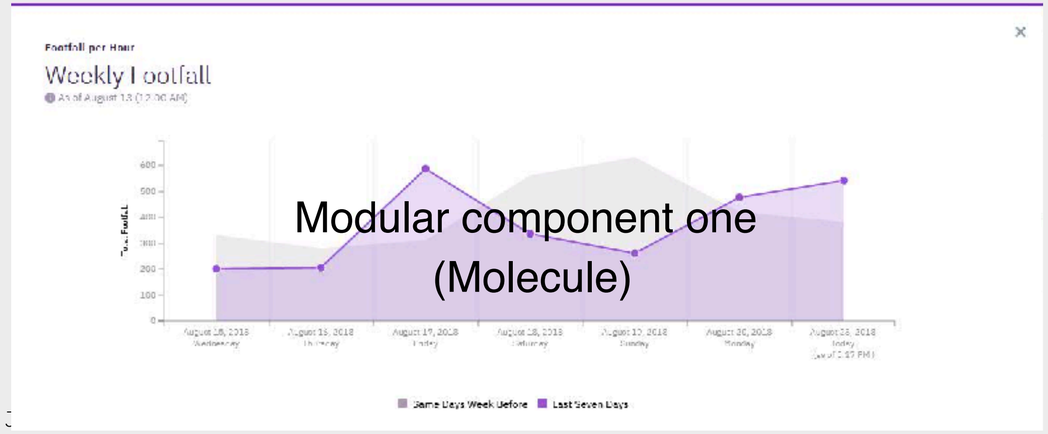
Assemble configured and styled components to create reusable units.

Component one (Atom)

+



=



Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Composite

Composite layer (Multi-Modular Components)

Molecular components in specific and validated combination to create page building blocks composed of display views + functional logic/service + styling or organisms.

For example:

Assemble configured and styled page building blocks to create reusable units.

Key for Layers

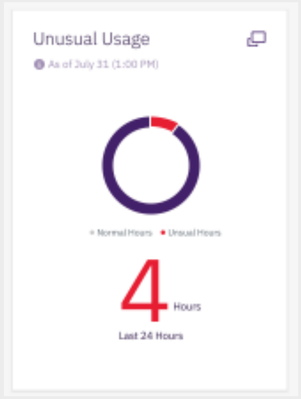
- Content Model
- Controller
- View Model
- View Particle
- View
- Composite

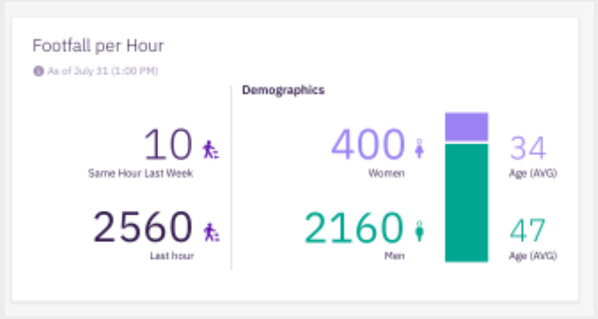
Composite layer (Multi-Modular Components)

Molecular components in specific and validated combination to create page building blocks composed of display views + functional logic/service + styling or organisms.

For example:

Assemble configured and styled page building blocks to create reusable units.



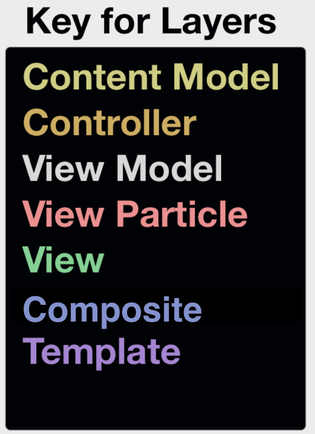


Key for Layers

Content Model
Controller
View Model
View Particle
View
Composite

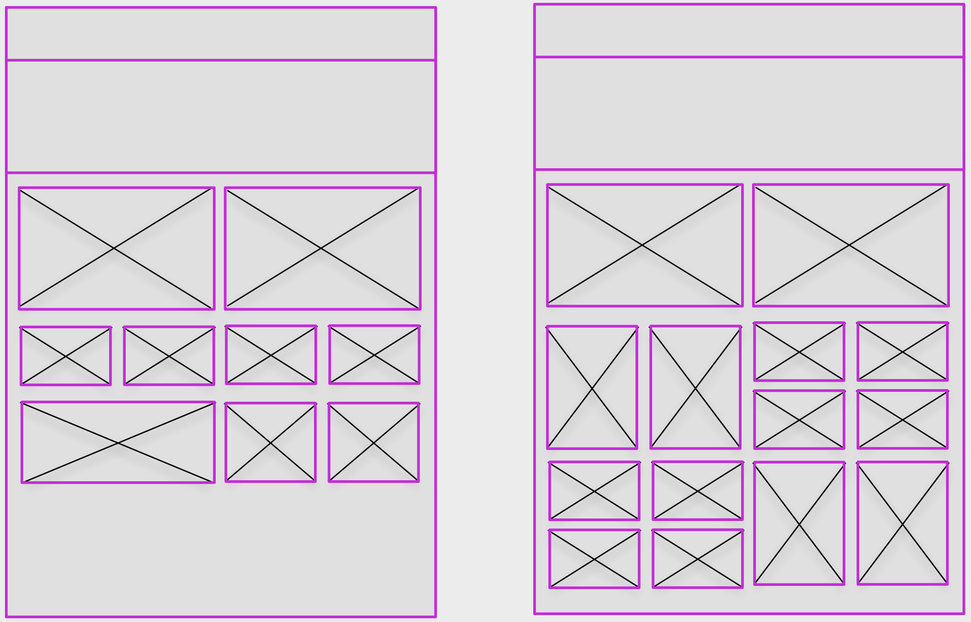
Template layer (Layout)

Defines the sections/grid/layout of the page, including responsive behaviour and contained components e.g. **Layout** and positioning of sections for Header, Navigation, UI Shell, sidebar, main content area, footer, etc.



Template layer (Layout)

Defines the sections/grid/layout of the page, including responsive behaviour and contained components e.g. **Layout** and positioning of sections for Header, Navigation, UI Shell, sidebar, main content area, footer, etc.



Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Composite
- Template

Page layer (Screen Instance)

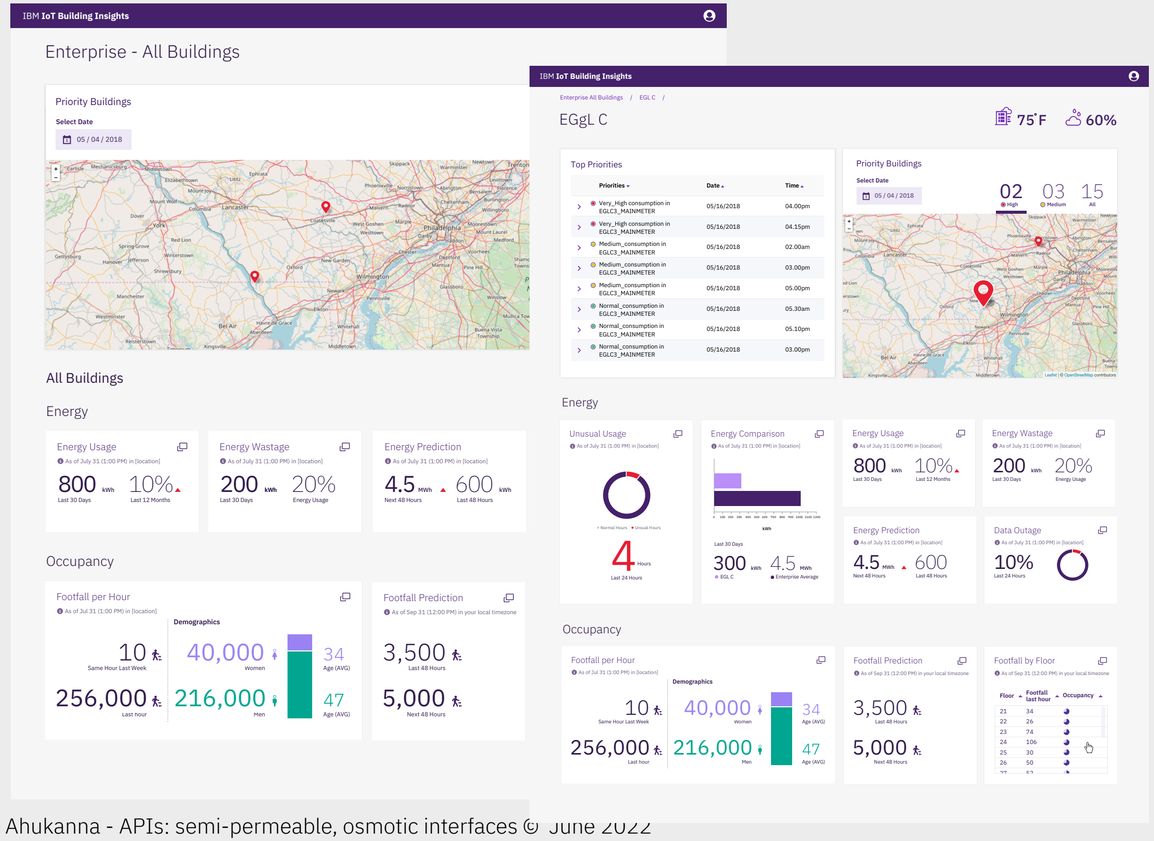
Aggregates, calculates and generates, in real-time, all the previous layers and displays the final resulting assembled page.

Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Composite
- Template
- Page

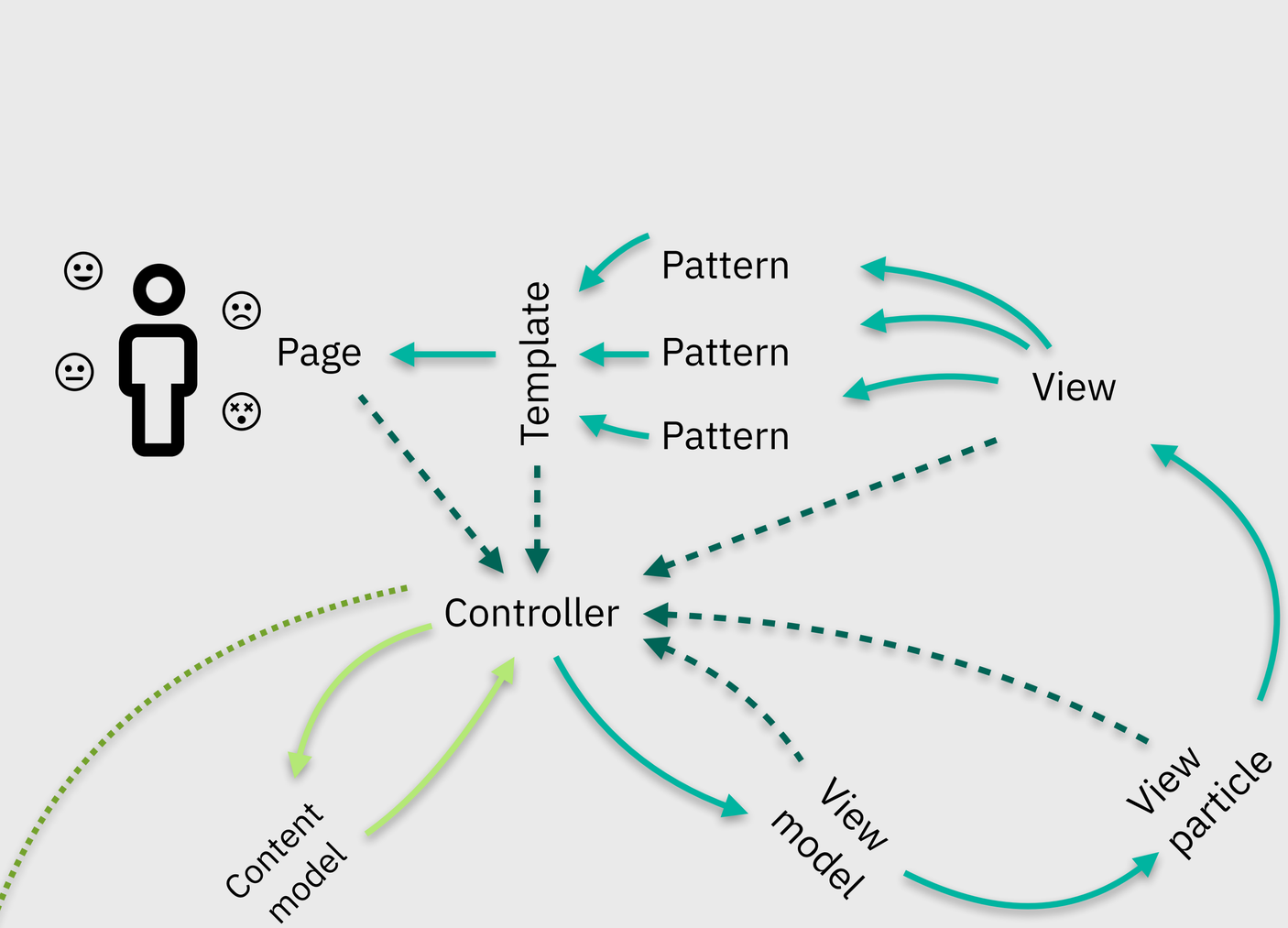
Page layer (Screen Instance)

Aggregates, calculates and generates, in real-time, all the previous layers and displays the final resulting assembled page.



Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Composite
- Template
- Page



External system integration API

Data integration API

App Interaction API

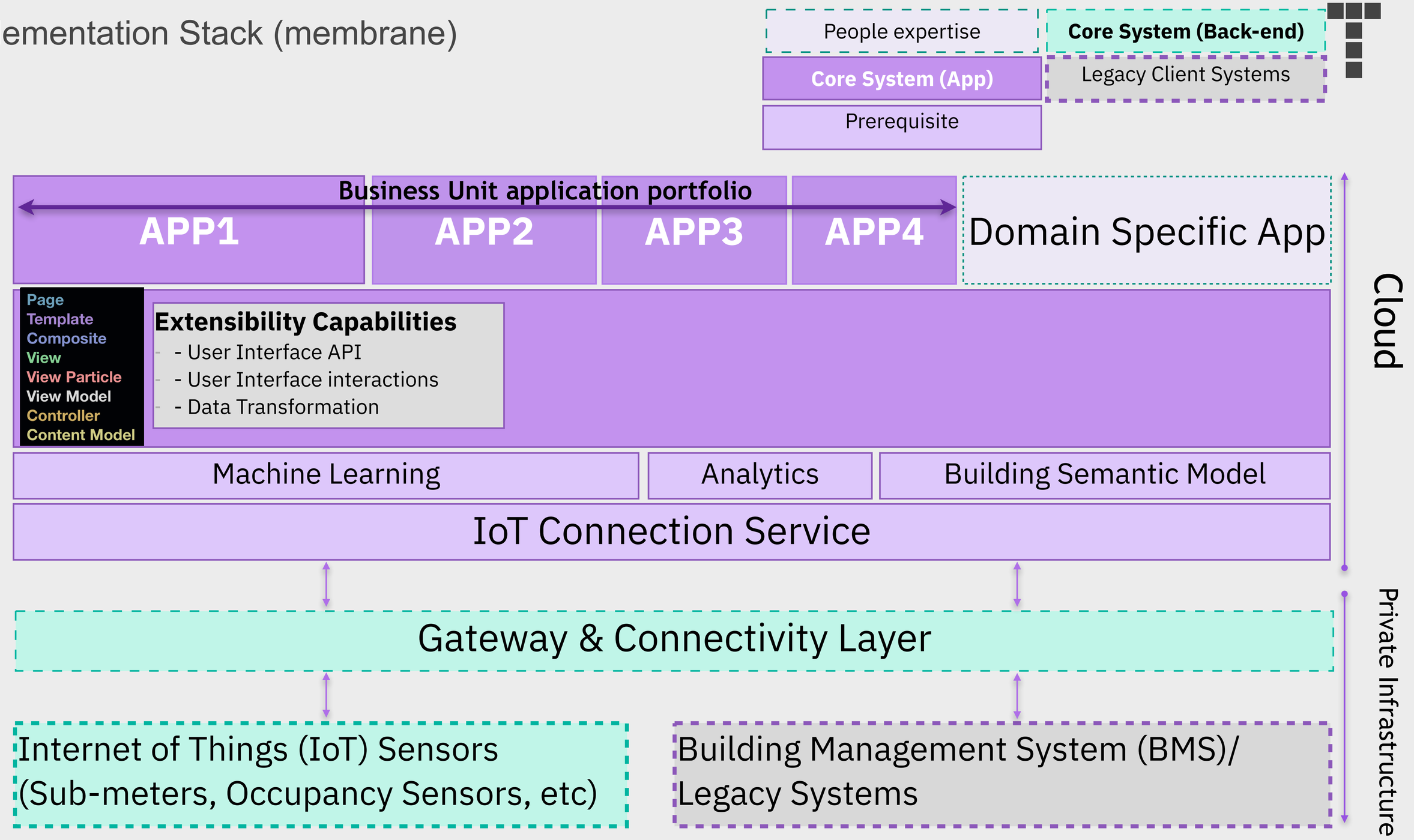
App Communication API

Key for Layers

- Content Model
- Controller
- View Model
- View Particle
- View
- Pattern
- Template
- Page

System Membrane Layers

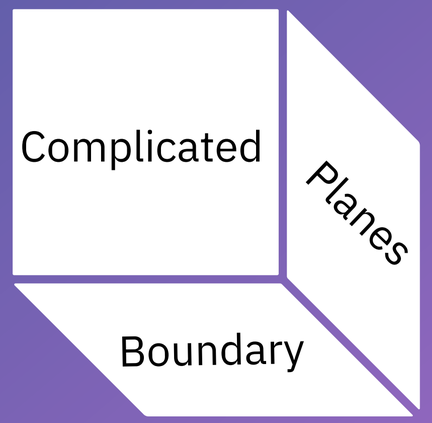
Implementation Stack (membrane)



Membrane Metaphors for API space

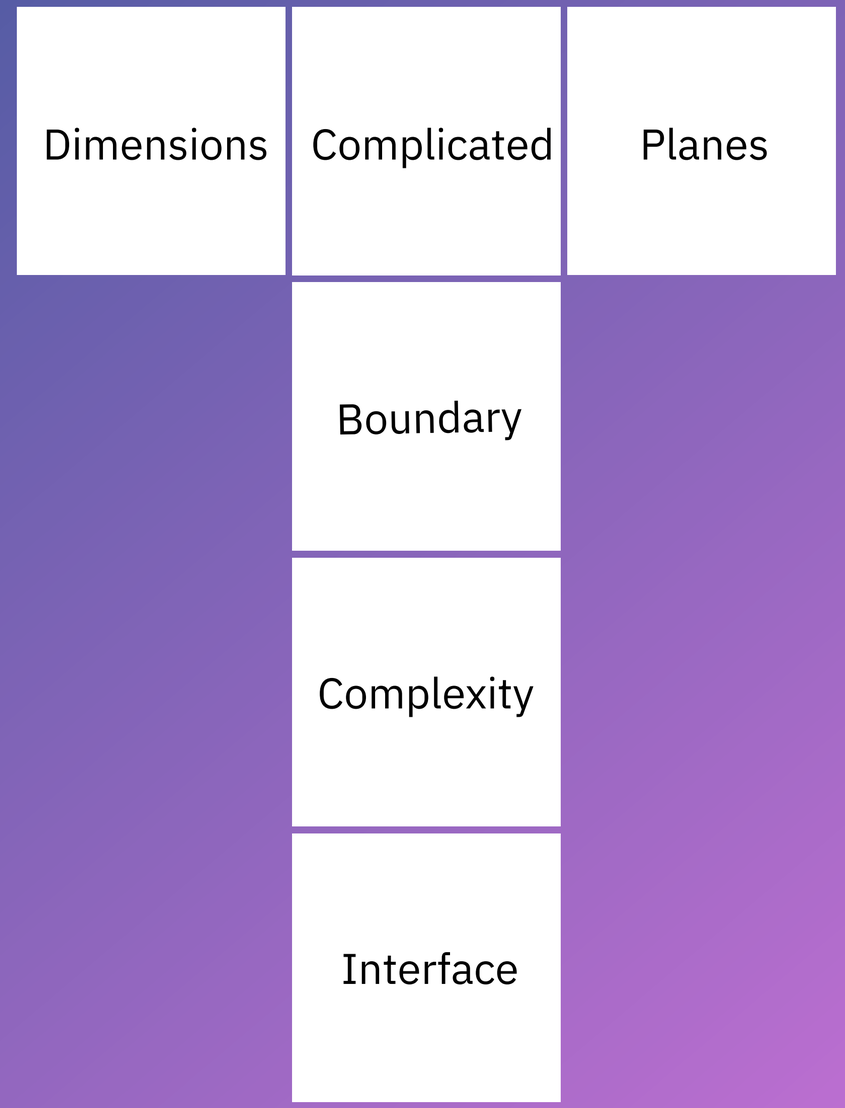
3D-cube

Think and model each challenge that your solving as a flattened 3-D space. Each flattened section is a semi-permeable membrane that only allows certain “molecule” objects to pass through.



3D-cube flat

Think and model each challenge that your solving as a flattened 3-D space. Each flattened section is a semi-permeable membrane that only allows certain “molecule” objects to pass through.



Q&A

Question: Do you see this being generic or frontend only, that is how do you position this with an API first approach?

Response:

Yes, I do think membrane approach is generally applicable but don't have the **empirical proof** to validate that suggested approach. See interaction flows on slide 3.

Only comment I'll make that API-First approach does not account for the contextual, relevant and viable interactions at higher interface levels. That's like giving a car chassis, engine, tyres, etc., leaving someone else to complete the assemble the functioning car.

Question: You seem to also take a boundary approach (Flores) but wondered if you've aware of Postel's law?

Response:

Yes familiar with postal’s law and it’s also a fundamental law of UX - https://lawsofux.com/postels-law/.

Question: I'm curious what inspired this approach, which category of models did you find inadequate or lacking?

Response:

I’m not aware of an existing software models that apply combinatorial cross- and mixed-disciplinary models to address solution challenges.

This approach has evolved and built on prior knowledge, experience and feedback from experiments.

Question: It did strike that is this a take on Parnas's encapsulation?

Response:

I have not heard of Parnas’s encapsulation. From a brief review, my membrane approach is more about only allowing certain matching objects to pass through based on contextual interactions e.g. osmosis is water flowing to balance concentration levels, moving from unbalanced concentration levels to balanced concentration levels.

Question: When we build such models and share it with the team who probably might just have a CS background, does it help to onboard the team?

Response:

I would share my initial model inspiration and transform/contextualise the model perspective to one they can relate to and apply. That's why I shared a practical, applied example from my daily work as validating empirical proof.

Comments and feedback:

- I always get instantly happy of Dawn, her aura and happy vibes.
- Good job Dawn. You just made a great ontological design in the API space!
- Thank you! super interesting concepts.
- Interesting. Chem in CS. Thank you.

Response:

Heartfelt thanks and glad you liked the presentation.