ARIEL UNIVERSITY

אוניברסיטת אריאל בשומרון

סדנת פרויקטים 2020

מפגש Design

22/1/20

יוסי זגורי
רכז פרוייקטים
052-4668866
Admin@Ariel.Zone

נושאי המפגש

- הודעות
- תזכורת פתיחת פרוייקט באתר GitHub
- עיצוב תוכנה: מושגים והגדרות
- ארכיקטורת תוכנה
- תבניות ארכיקטוניות
- שלב עיצוב – המלצות, פעילויות ותוצרים
- מה הלאה

הודעות

- ניתן להכנס לאתר ולהחליף תיבת מייל (אם הנוכחית אינה בשימוש) וסיסמא (מומלץ).
- נא לדאוג להזין את הלינק לאתר GitHub עד סוף השבוע (ב- Ariel.zone) - מספיק ע"י אחד מחברי הקבוצה.
- באתר ה- GitHub יש להעלות את שני התוצרים עד עתה ולדאוג לתאור מסודר וציון שמות חברי הקבוצה.
- יש לדאוג מול המנחה להזנת ציונים לשלבים המקדים וגיבוש הדרישות - **אופציה זו ננעלת ב- 31.1 !**

היכן אנחנו

פיתוח תוכנה הינו ממלכת אי וודאות
אי ודאות : אנושית, פיננסית, טכנולוגית, תהליכית וכ'
בכל שלב אנו מקלפים נתח מאי הוודאות ומקבלים החלטות מיודעות

<u>מרחב הבעיה</u>
הכרה, הבנה, ניתוח
שלב החזון, הכרה ,ניתוח ואיפיון דרישות מהפתרון

<u>הפתרון</u>
עיצוב, פיתוח, בדיקות, הטמעה

האינטלקט האנושי חותך את המציאות לפיסות וחלקים, יוצר גבולות
ומסווג פיסות אלה ברמות אבסטקרציה משתנות

What is Design?

Design is a plan or construction of a system, satisfying goals and constraints

Software Design

<=>

Modulation

<=>

Dependency Management

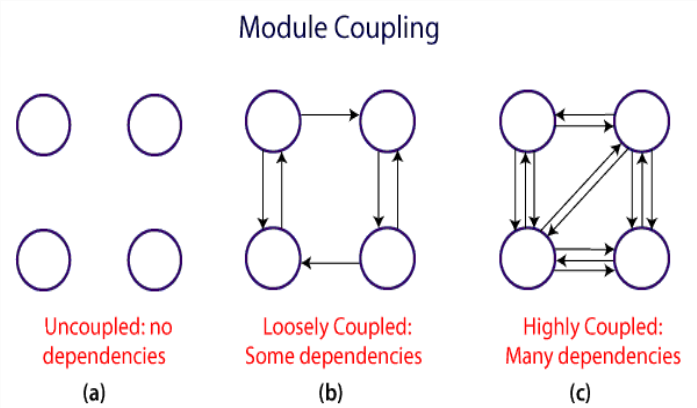
What Is Modularity?

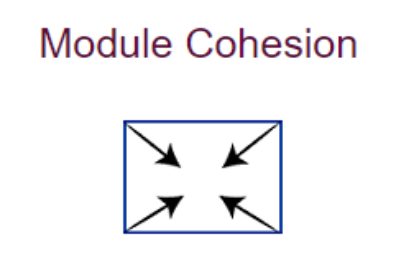
- The degree to which a system's components may be separated and recombined
- Primarily used to reduce complexity
- Benefits: flexibility and variety in use
- Breaking system to set of Parts
- Parts hiding complexity behind an abstract interface

Measures: Coupling & Cohesion

Coupling & Cohesion

Coupling	Cohesion
Coupling is also called Inter-Module Binding.	Cohesion is also called Intra-Module Binding.
Coupling shows the relationships between modules.	Cohesion shows the relationship within the module.
Coupling shows the relative **independence** betwe en the modules.	Cohesion shows the module's relative **functional** strength.
While creating, you should aim for low coupling, i.e., dependency among modules should be less.	While creating you should aim for high cohesion, i.e., a cohesive component/ module focuses on a single function (i.e., single-mindedness) with little interaction with other modules of the system.
In coupling, modules are linked to the other modules.	In cohesion, the module focuses on a single thing.



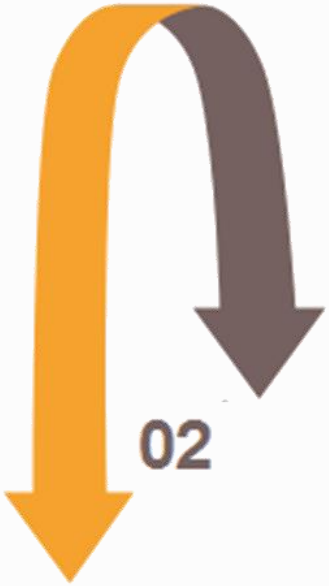


Design Levels

High Level

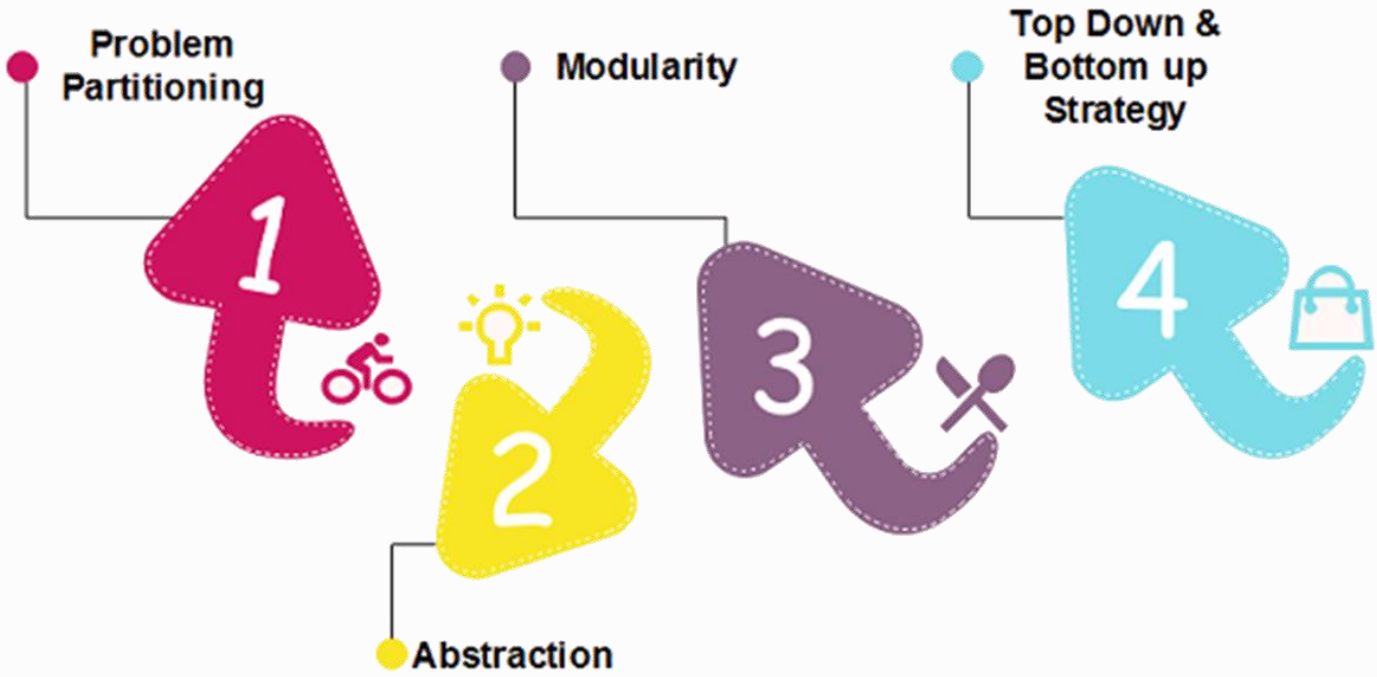
01 Focus on deciding which modules are needed, their specification, relationship and interface among them

Low Level



02 Focus on planning and specifying the internal structure and processing detail

Design Strategy

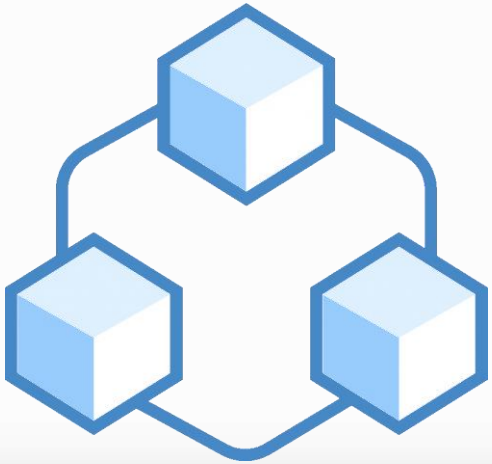


Design Strategy II

- ***Use Models*** to analyze and reduce risk
- ***Use models*** and visualizations as communication & collaboration tool
- ***Identify*** key (architectural) engineering decisions
- ***Build to change*** instead of building to last.

Terms

- Module → "Part" of the solution
- Component → a **software** unit with a well-defined interface and explicitly specified dependencies.
- Logical or physical : Software Package, Web Service ….
  - Layer – logical Module having DAG Dependency structure
  - Tier – Physical Place representing unit of deployment , where "code runs"



What is architecture?

"Architecture is a word we use when we want to talk about <u>Design</u> but want to puff it up to make it sound important"

Who Needs an Architect, Martin Fowler

What is architecture?

"Architecture is about the important stuff. Whatever that is".

Prof. Ralph Johnson
GOF Member

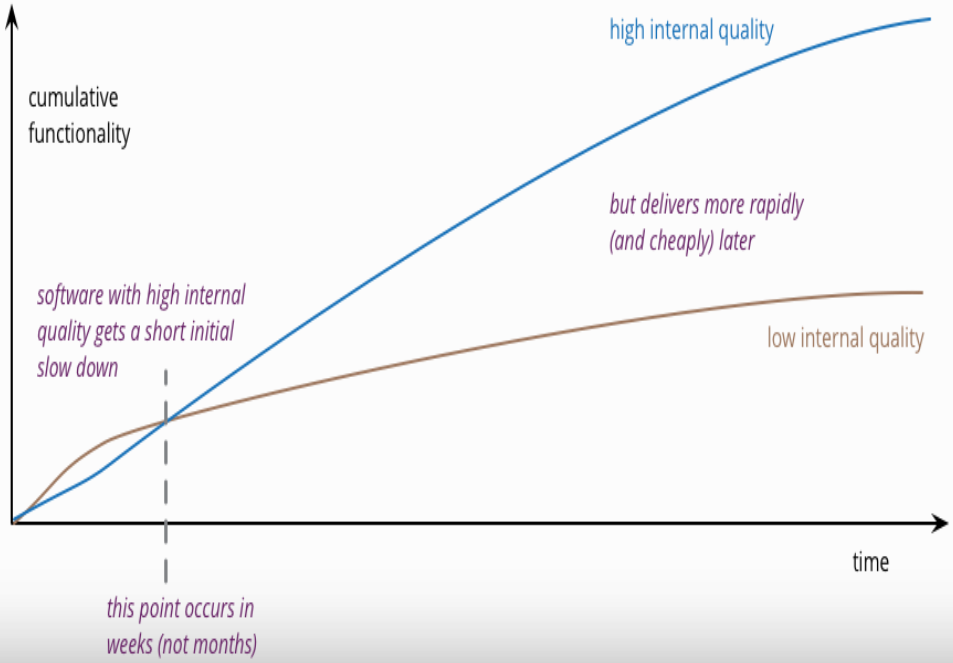
- Thinking architecturally => **Deciding** what is important
- Important=>**Recognizing** what elements are likely to result in serious problems should they not be controlled
- **Keeping** those architectural elements in good condition

Bad architecture lead to difficulty to modify, in turn leading to features added more slowly and with more defects

Cost of good architecture

- Tricky subject for customers and users of software products - as it isn't something they immediately perceive

- Subjectively, investing in good architecture “costs” only several weeks, not months



Architectural Considerations

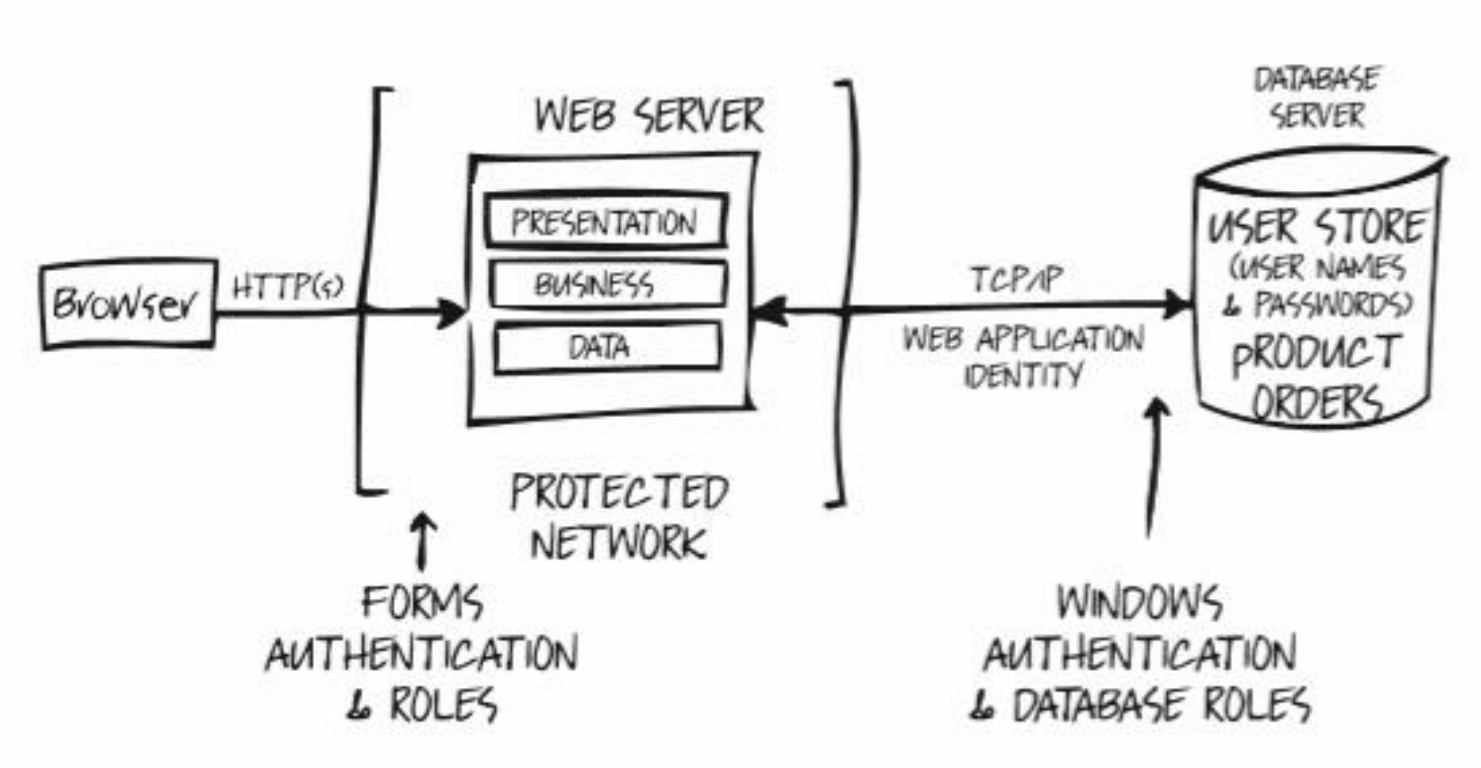
- What are the foundational parts of the architecture that represent the greatest risk if you get them wrong?
- What are the parts in the design are most likely to change, or whose design you can delay until later with little impact?
- What are your key assumptions, how will you test them?
- What conditions may require you to refactor the design?

Do not attempt to over engineer the architecture, and do not make assumptions that you cannot verify. Instead, keep your options open for future change

Recommended Approach

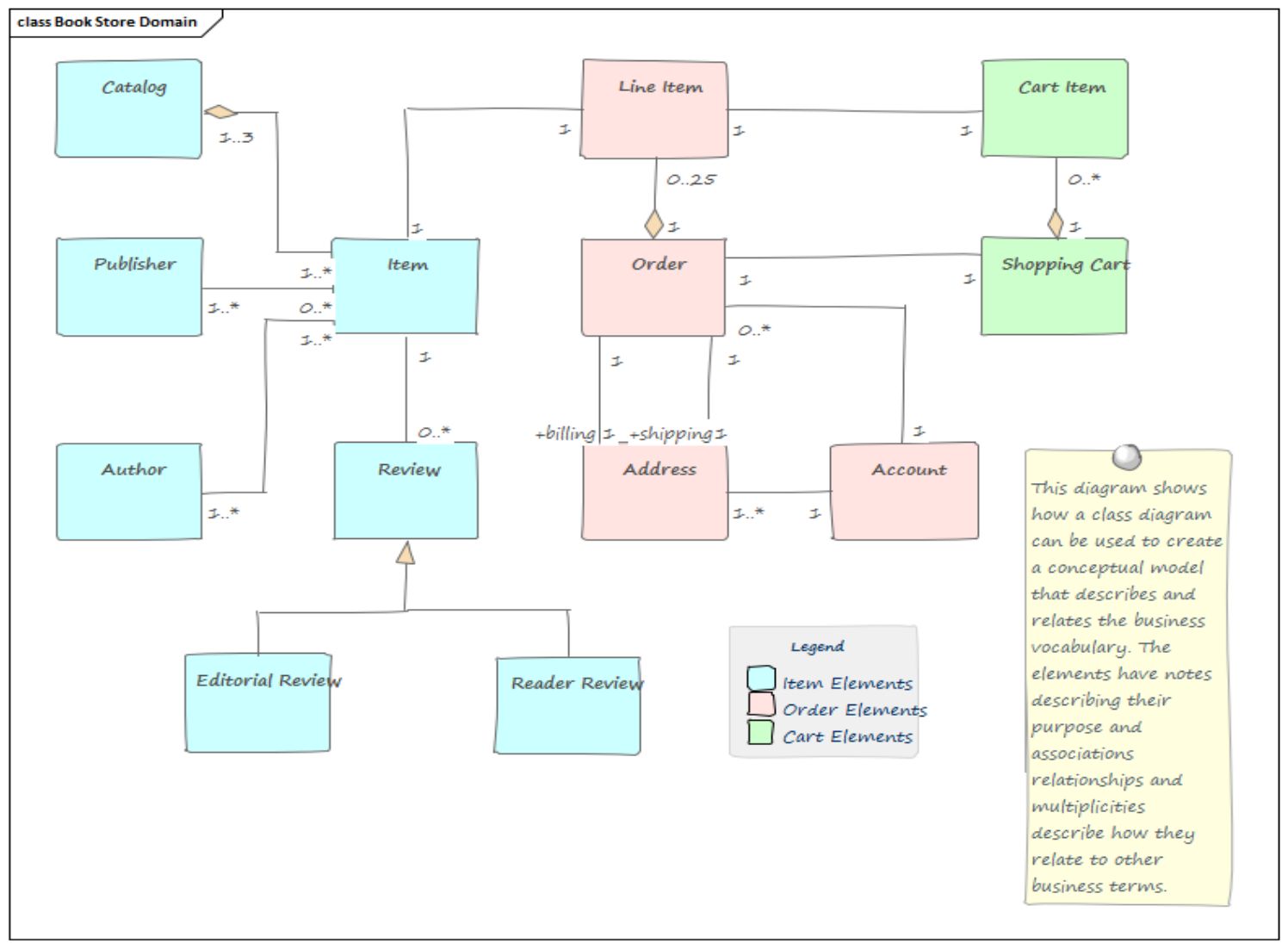
- Define User Types (Roles/Actors)
- Define Services for each User type
- Define User Process Workflows (UW)
- Sketch out user interface for each UW
- Communicate Sketches with Stakeholders to get approval
- Extract Data profile managed by the System
- Place logic according to selected architectural style / selected patterns

Whiteboarding



Conceptual Model

Example



Stage Artifacts

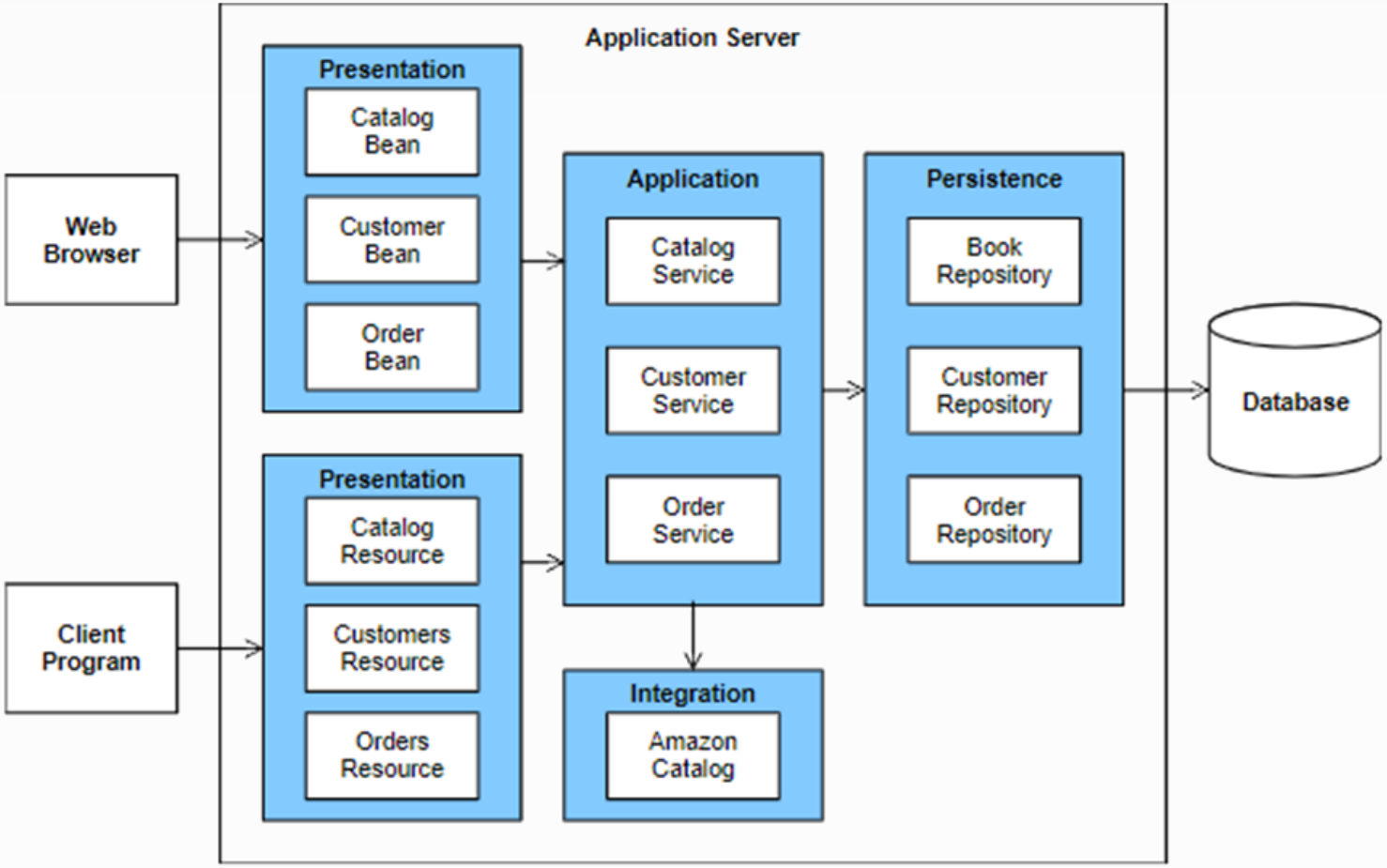
- High Level Design
- Identify the **MAIN** modules & concerns that would be developed for the system and their interfaces
- Provides a view of the system at an **abstract** level
- Describe the hardware and software you will use to develop the application
- How application will work=> <u>Low</u> Level Design

Stage Artifacts

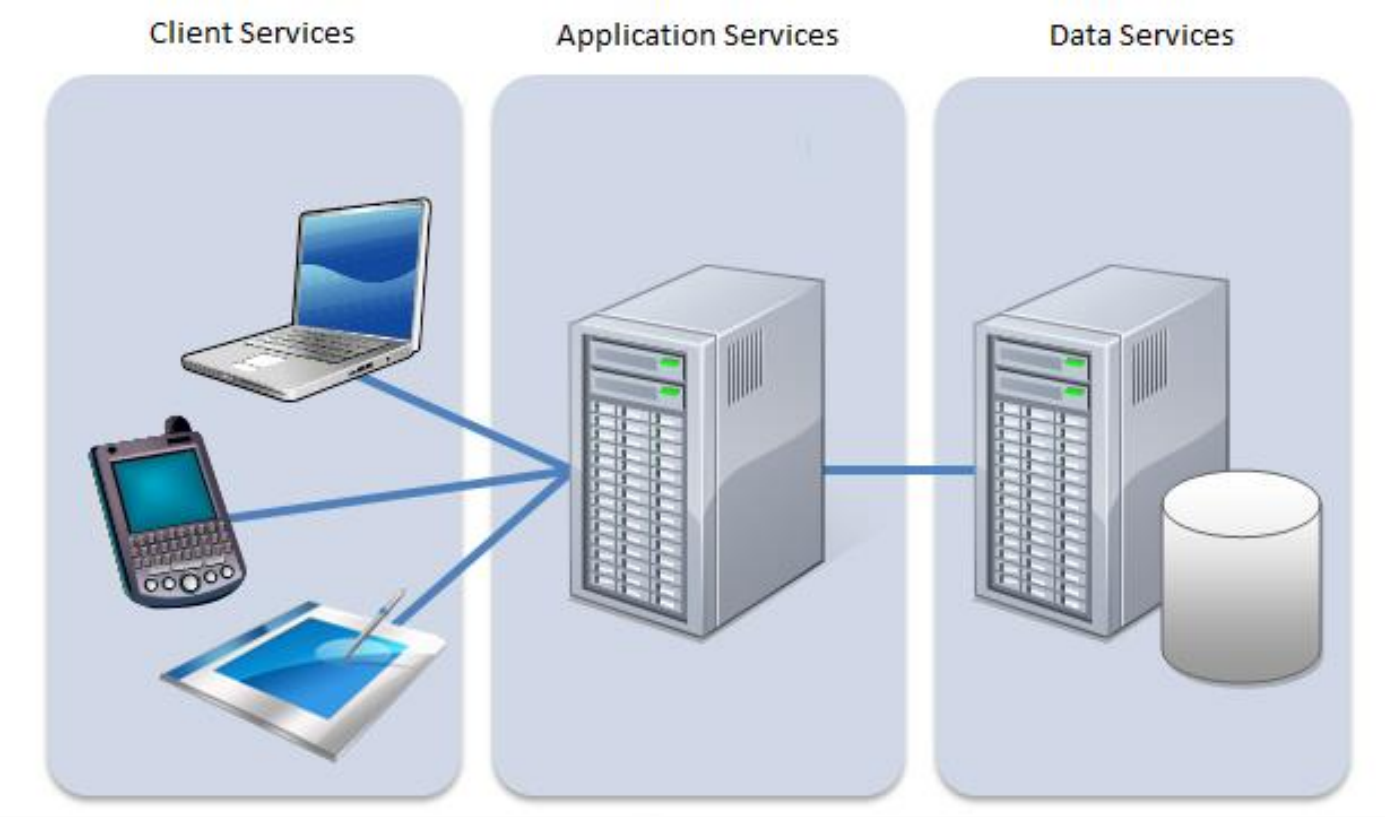
- Software Design Document (SDD)
- Functional Modulation Block Diagram
- System Modulation Block Diagram
- Domain / Conceptual Model (E/R Diagram)

Modulation Block Diagram

Example



Solution Tiers Diagram



ARIEL UNIVERSITY

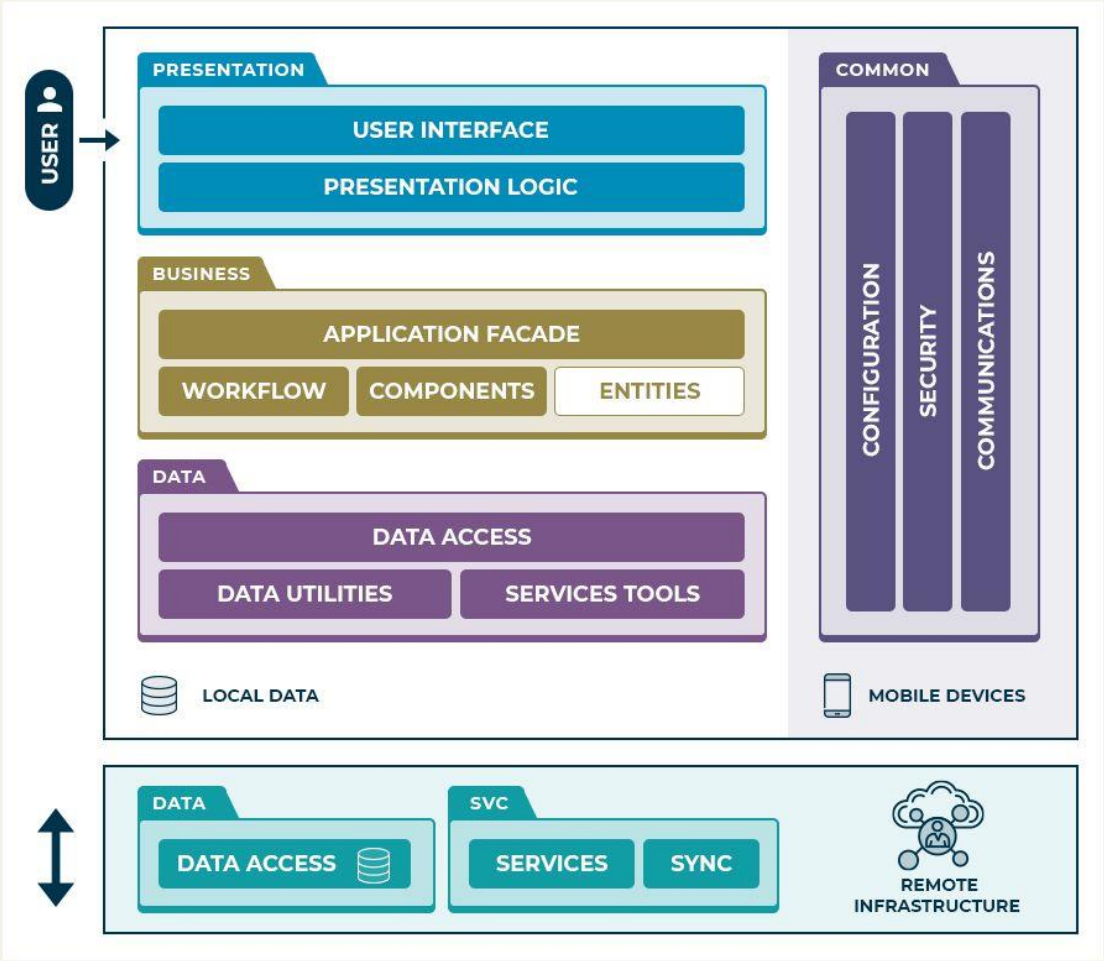
אוניברסיטת אריאל בשומרון

Application Architectural styles

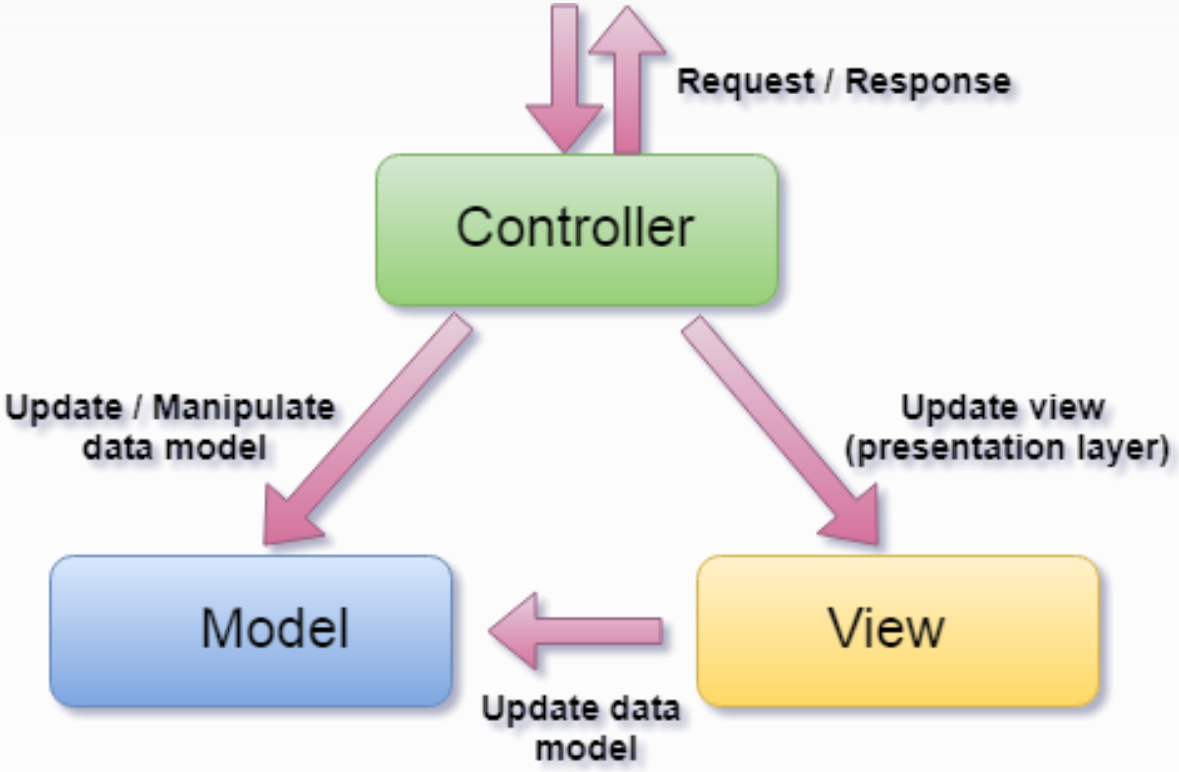
Selected Topics

Common Modules

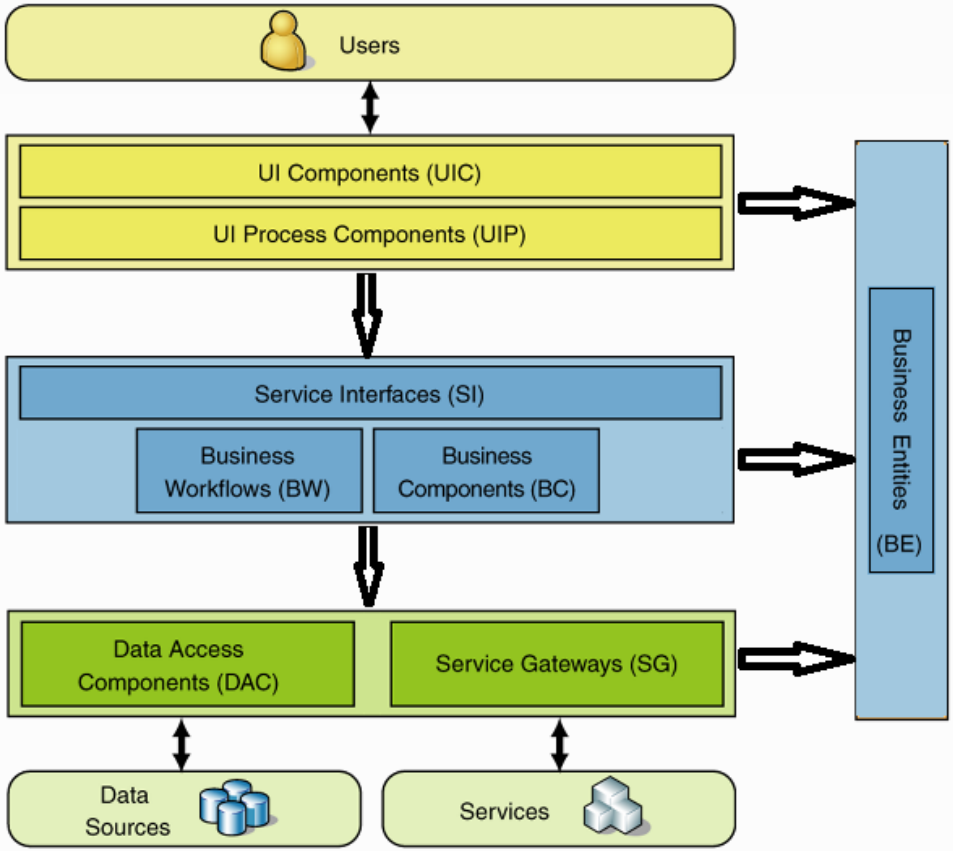
Partitions & Responsivities



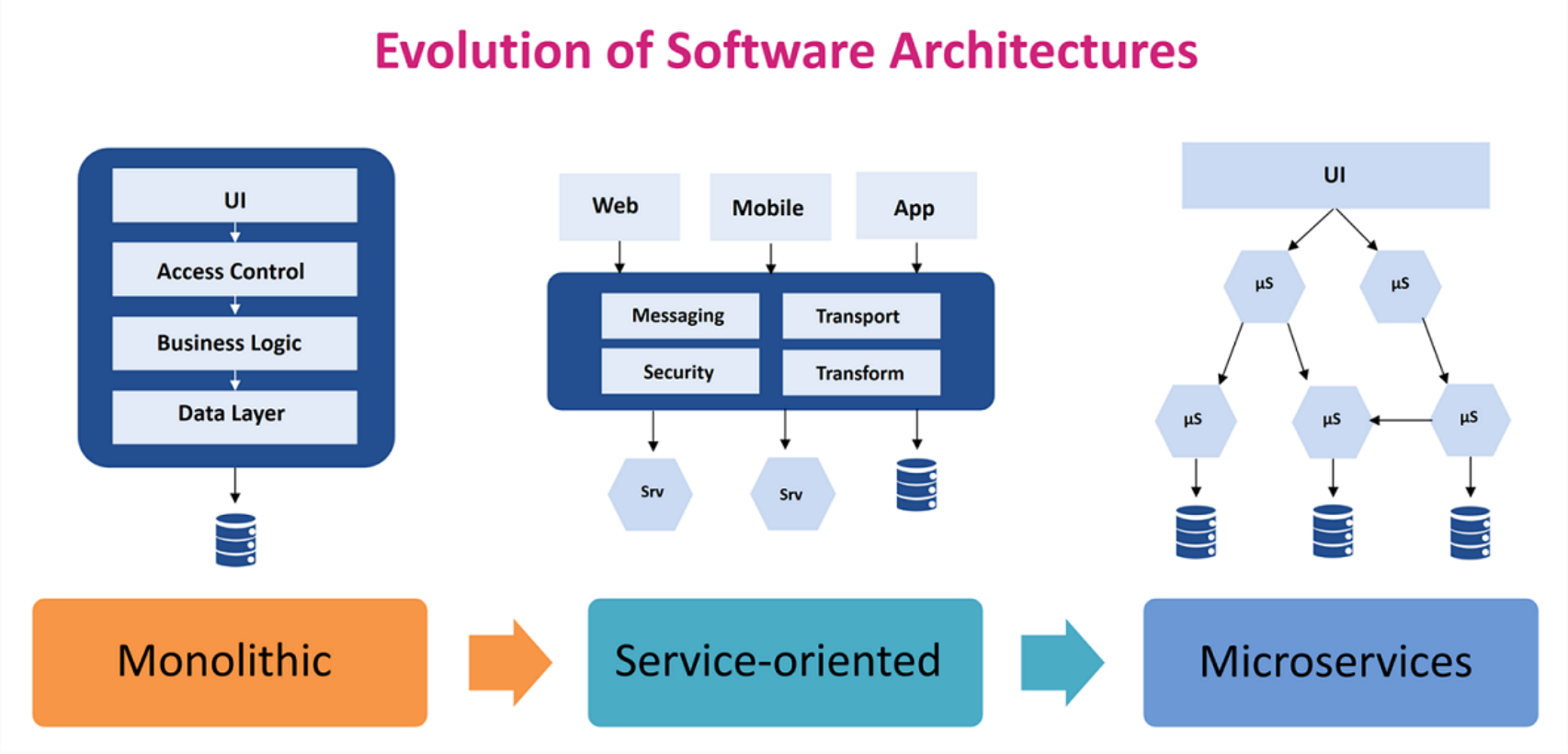
MVC


Request / Response
Controller
Update / Manipulate
data model
Update view
(presentation layer)
Model
View
Update data
model

Multi Layer Pattern



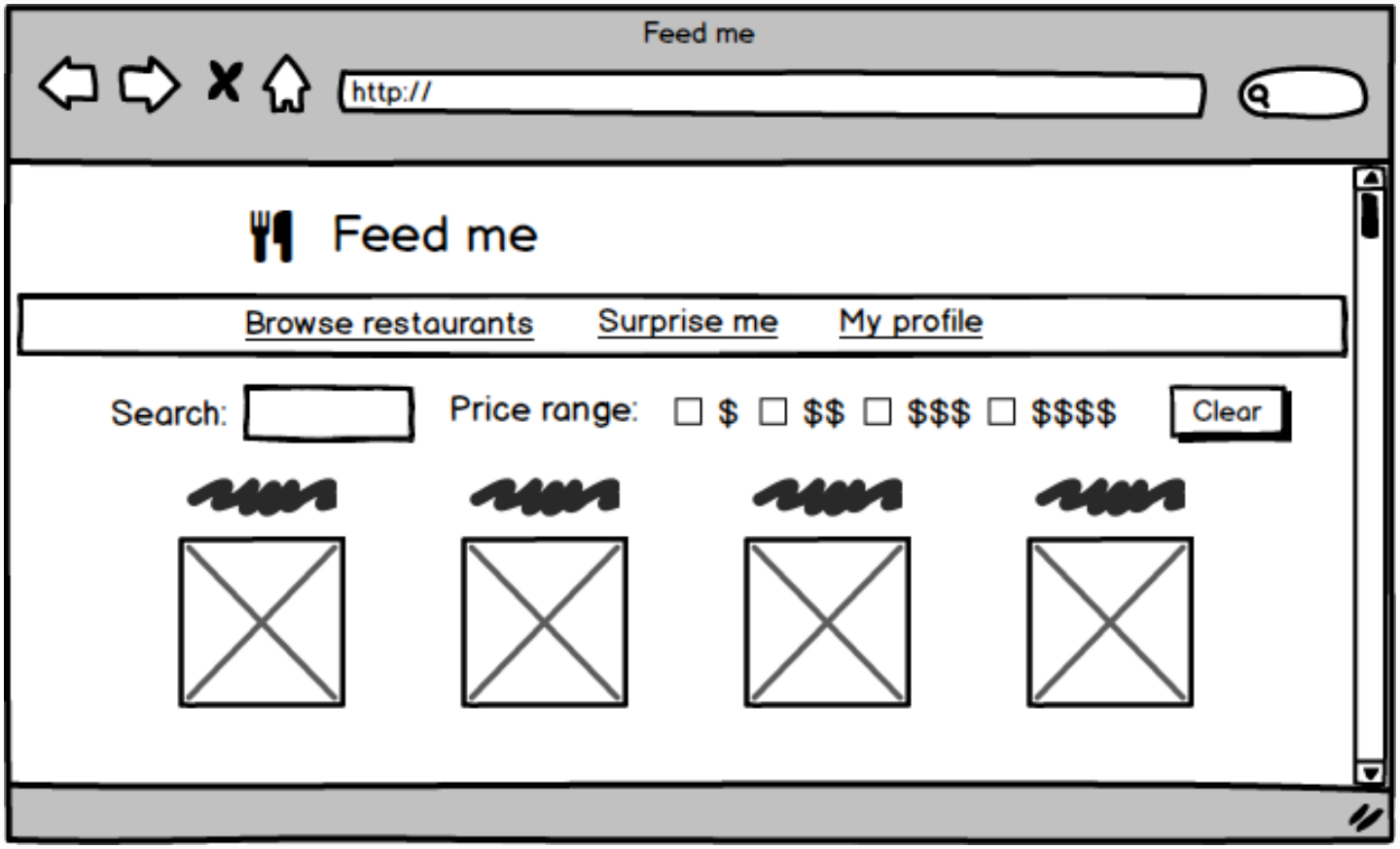
Braking the Monolith



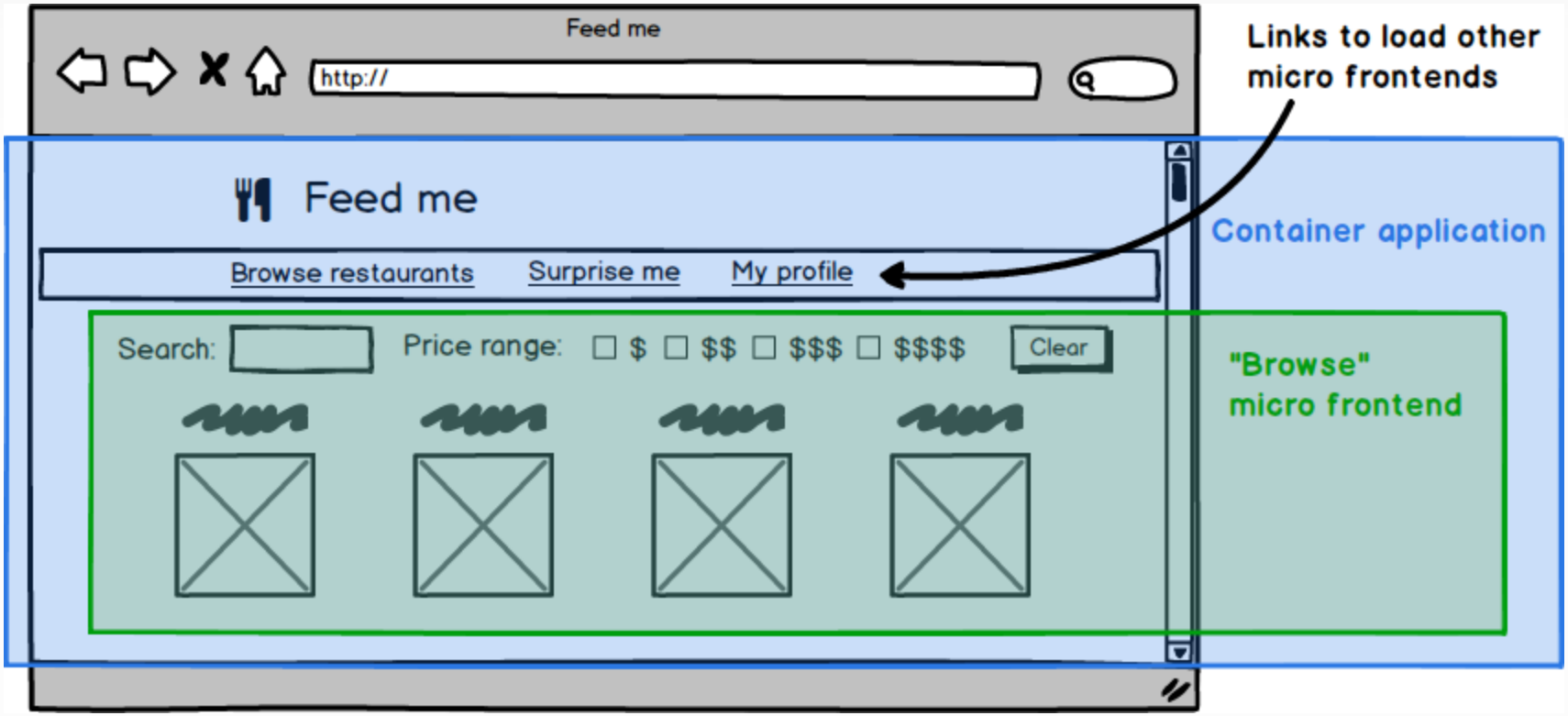
Consider

- Services
- Micro Services
- Micro Frontends
- Serverless
- Docker (To Support CI/CD)

Micro Frontends



Micro Frontends



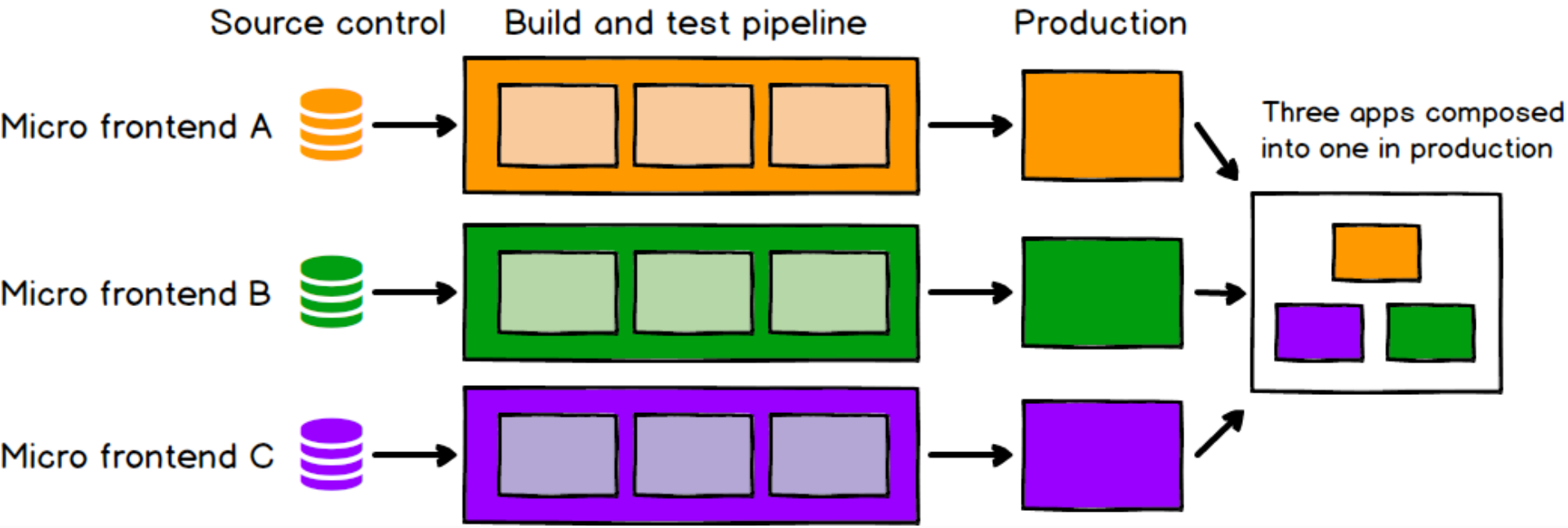
- Renders common page elements such as headers and footers
- Addresses cross-cutting concerns like authentication and navigation
- Brings the various micro frontends together onto the page, and tells each micro frontend when and where to render itself

Micro Frontends

- *Same Rational as Micro Services as well as ANY Design approach*
- *Breaking up frontend monoliths into many smaller, more manageable pieces*
- Simpler chunks that can be developed, tested and deployed independently, while still appearing to customers as a single cohesive product.
- Smaller, more cohesive and maintainable codebases
- More scalable organizations with decoupled, autonomous teams
- Increase ability to upgrade, update, or even rewrite parts of the frontend in a more incremental fashion than was previously possible

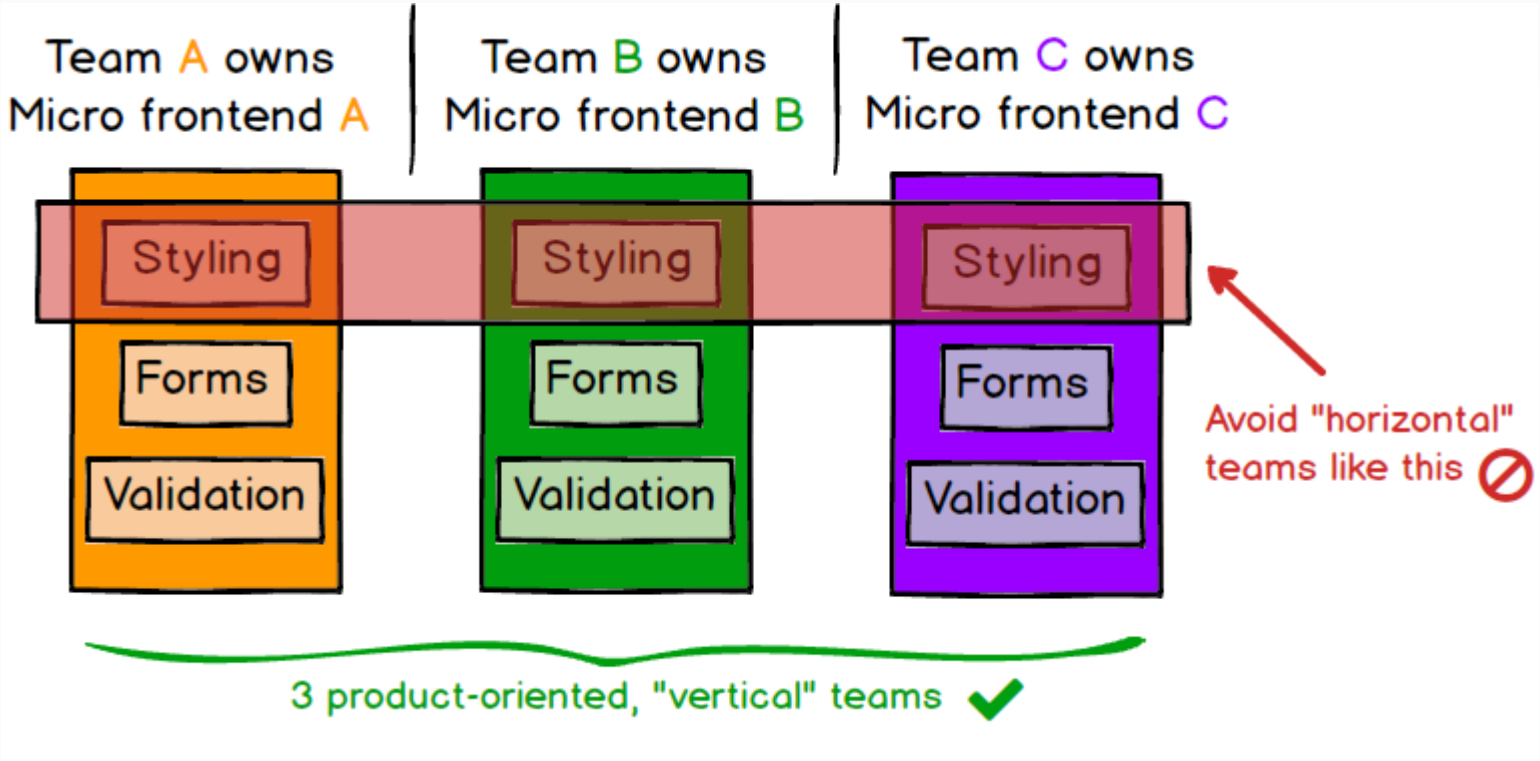
"A Design style where independently deliverable frontend applications are composed into a greater whole"

Micro Frontends



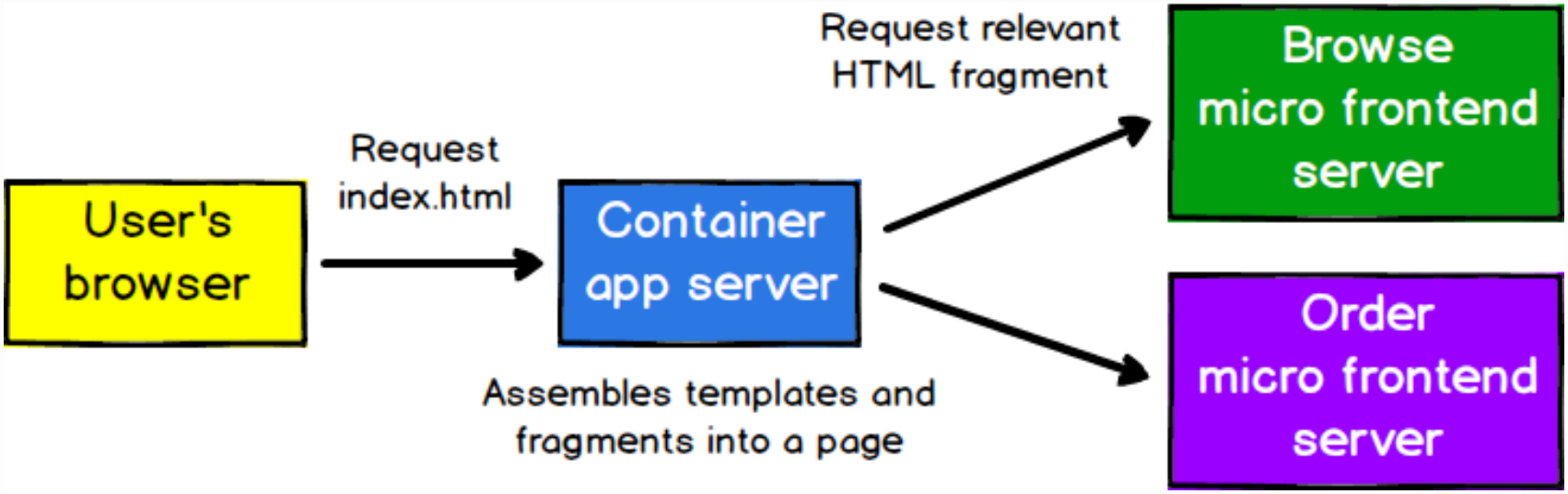
Micro Frontends

Major Best Practice



Micro Frontends

Server-side template composition



Optional Design Elements & Services

- Message Brokers
- Rule Engines
- Workflow Engines
- …

מה הלאה

- השלב הנוכחי הינו שלב עיצוב עילי (HLD)
  - מועד אחרון להגשת מסמך SDD : 17.3.20
  - עם סיומו והערכת המנחה יש להעלותו לאתר הפרוייקט ב GitHub
- המשך מפגש כל שבועיים עם המנחה לבחינת סטטוס התקדמות.
- לא להמתין עם התחלת פיתוח - להתחיל אחרי השלמת "עיצוב מספק".
- חלוקת תפקידים והגדרת אחריות לכל חבר צוות.
- מפגש שלב הבא : 18.3.20
- השתמשו באתר הסדנה **והכנסו אליו תכופות**
- ניתן ליצור עימי קשר במייל (admin@ariel.zone)או להגיע לשעת קבלה בצמתי קבלת החלטות

בהצלחה!