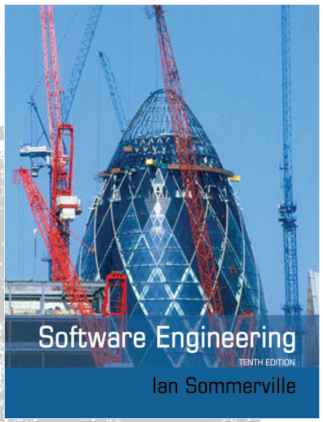
Fundamentals in Software engineering Lesson 3



יסודות בהנדסת תוכנה שיעור 3

Dr. Hadas Schwartz-Chassidim

October 2021

בהכנת המצגות בקורס נעזרתי בספרות ובמצגות של פרופ' יאן סומרסוויל

http://iansommerville.com/software-engineering-book/

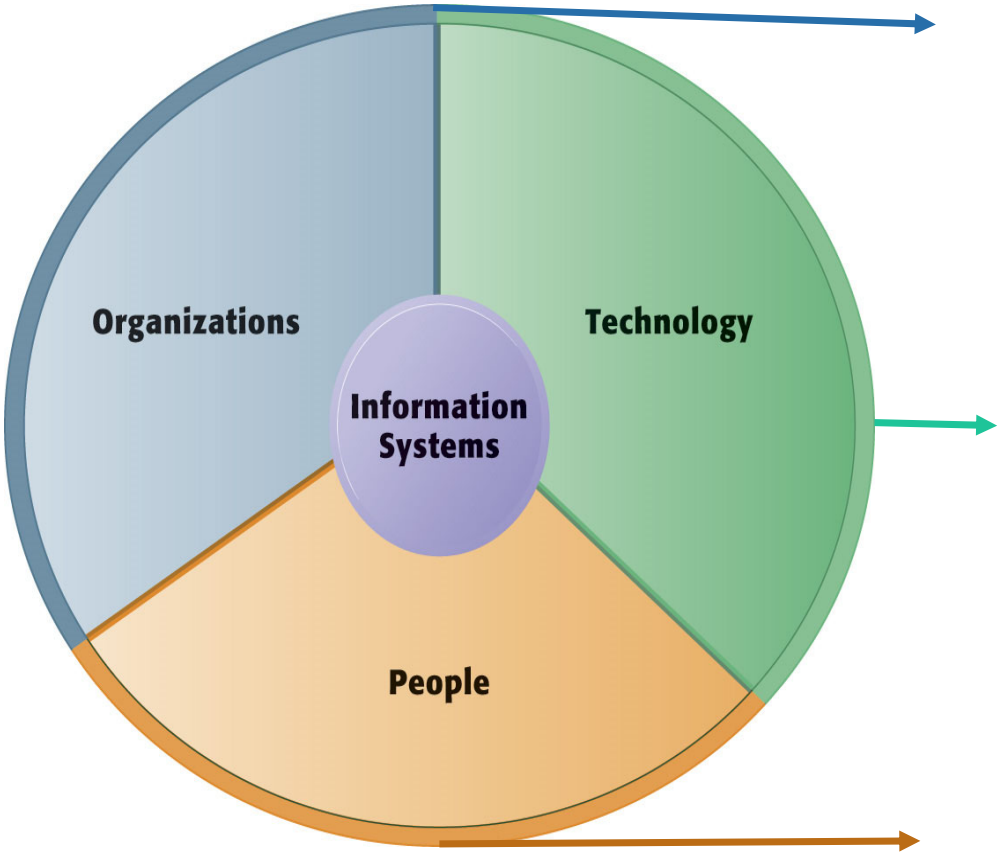
LESSON TOPICS

- Design processes
- SDLC review of prominent methodologies
- Architecture design
- Handling changes Agile
  - Mob programing

A good requirement

- Correct
- Unambiguous (all statements have exactly one interpretation)
- Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
- Consistent
- Ranked for importance and/or stability
- Verifiable (avoid soft descriptions like “works well”) -testable
- Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
- Does not specify any particular design
- Traceable (cross-reference with source documents and spawned documents).

Who writes the requirements



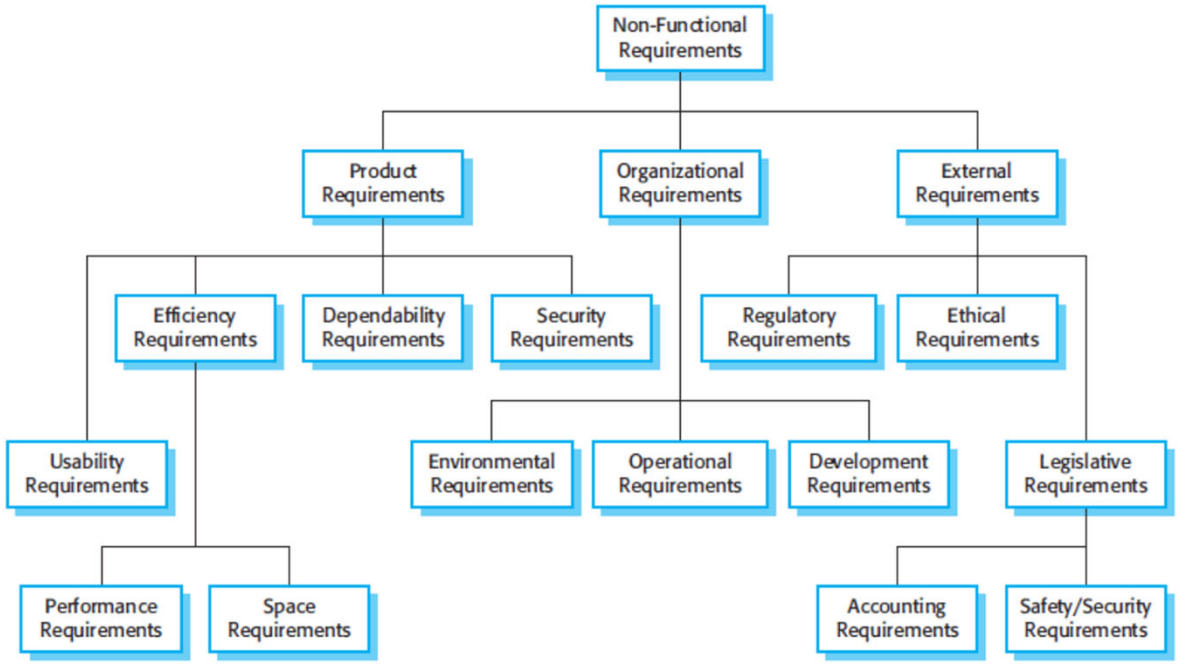
System Customers	Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers	Use the requirements to understand what system is to be developed.
System Test Engineers	Use the requirements to develop validation tests for the system.
System Maintenance Engineers	Use the requirements to understand the system and the relationships between its parts.

Functional and non-functional requirements

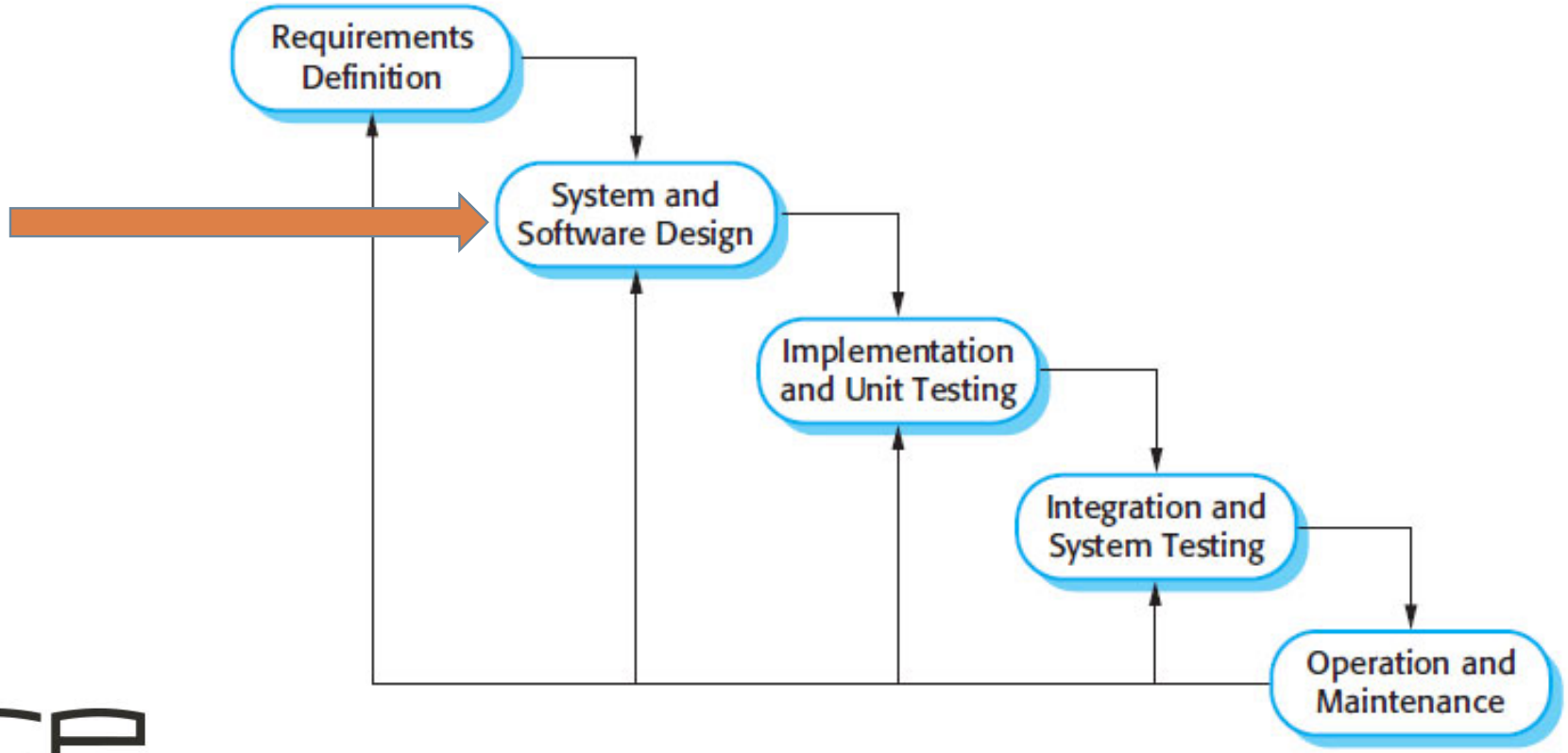
- **Functional:** The statements of services the system should provide, how the system should react to particular inputs
- **Non functional:** The constraints on the services or functions offered by the system (time, operational, performance, speed, load)

The distinction between functional and non functional is not Clear cut

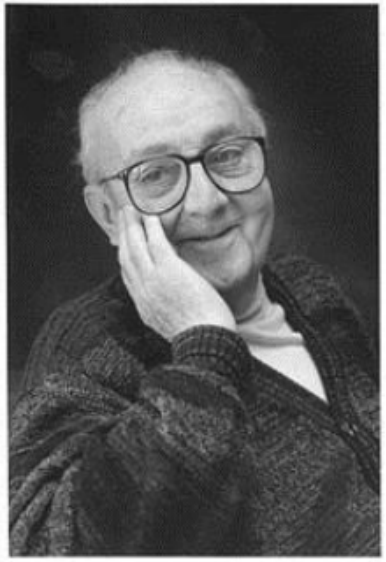
Non-Functional Requirements



Waterfall model



Models…



"ALL MODELS ARE WRONG,
BUT SOME ARE USEFUL"

George E. P. Box, 1919-2013

Context and boundaries

Context and boundaries

- Relations with the environment
- System boundaries
- Business processes and **computer processes** (e.g., budget plan, ordering, sale).
  - Input (origin, format, platform, device, interface)
  - Output (destination, format, platform, device, interface)
  - Process (update, cancel, manipulation)
  - Search
  - Users
  - Trigger
  - Frequency
  - Logic

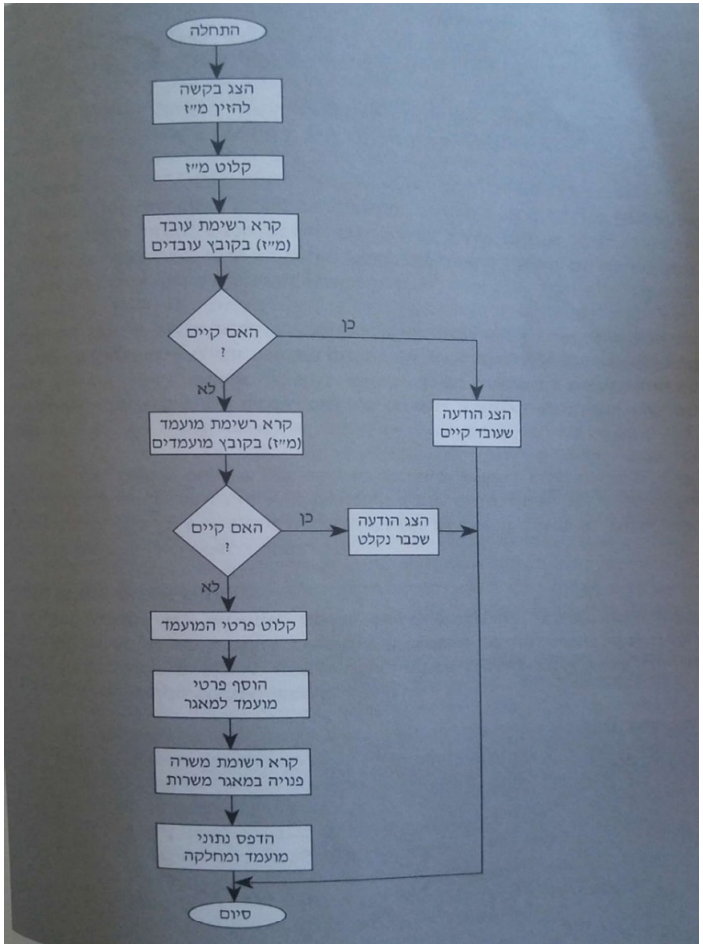
Manual + Computer processes

- מחלקת משאבי אנוש מגייסת עובדים חדשים לחברת הייטק, פתיחת סניף חדש
- אחת הפעולות היא רישום מועמדים חדשים לעבודה
- מועמד פונה, מופנה למשרדי החברה (HR), ממלא טופס, הטופס נבדק לפני קליטה, אם תקין ומתאים, מוזמן לראיון עבודה.
- **באיזה היקף המחשב מבצע את הפעולות?**
- **נקודות בדיקה?**

Manual + Computer processes

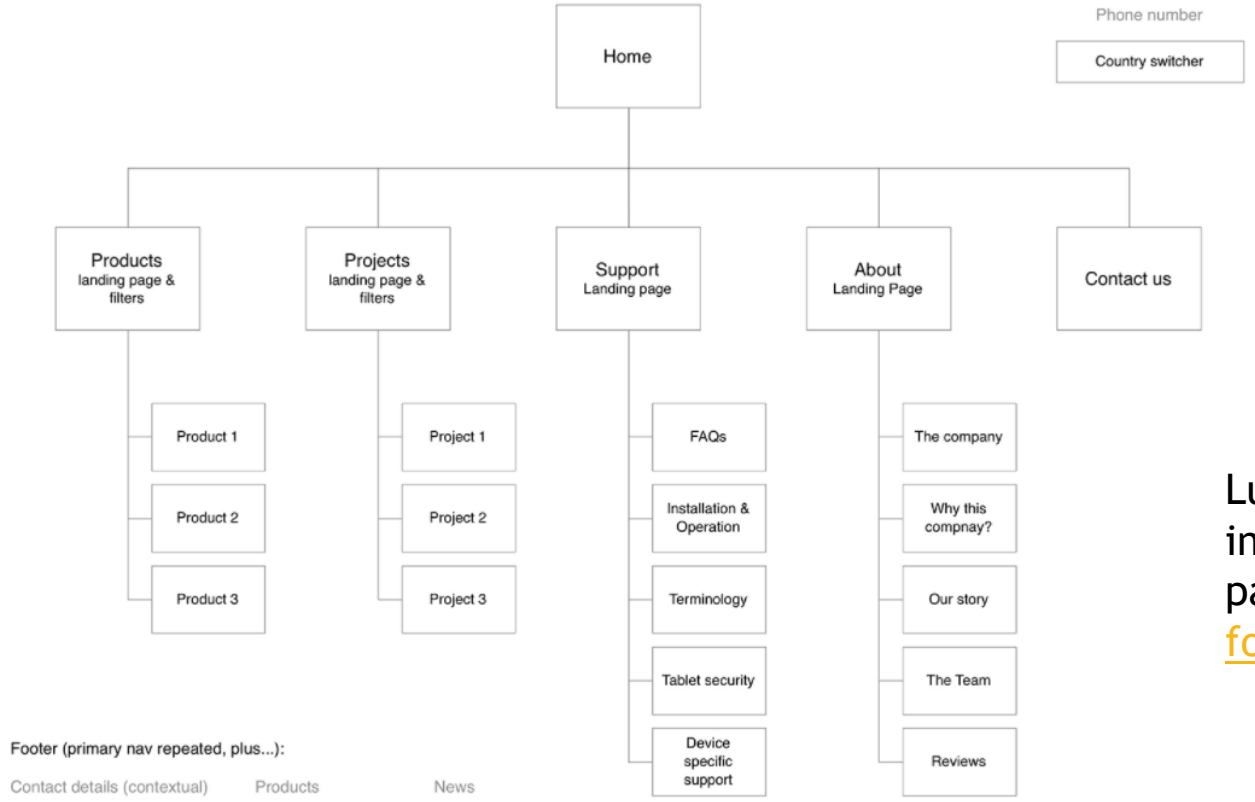
- מילוי ידני
- ממוחשב
  - בדיקה האם העובד קיים?
  - אם קיים --- > סיום
  - לא קיים הזנת פרטי מועמד במערכת כ"א
  - בדיקת פרטים ותקינותם
  - הפנייה למחלקה המגייסת, הודעה למנהל המחלקה לצורך תיאום ראיון
- תיאור כללי - יותר פירושים ווריאציות יישום

Process flow diagram



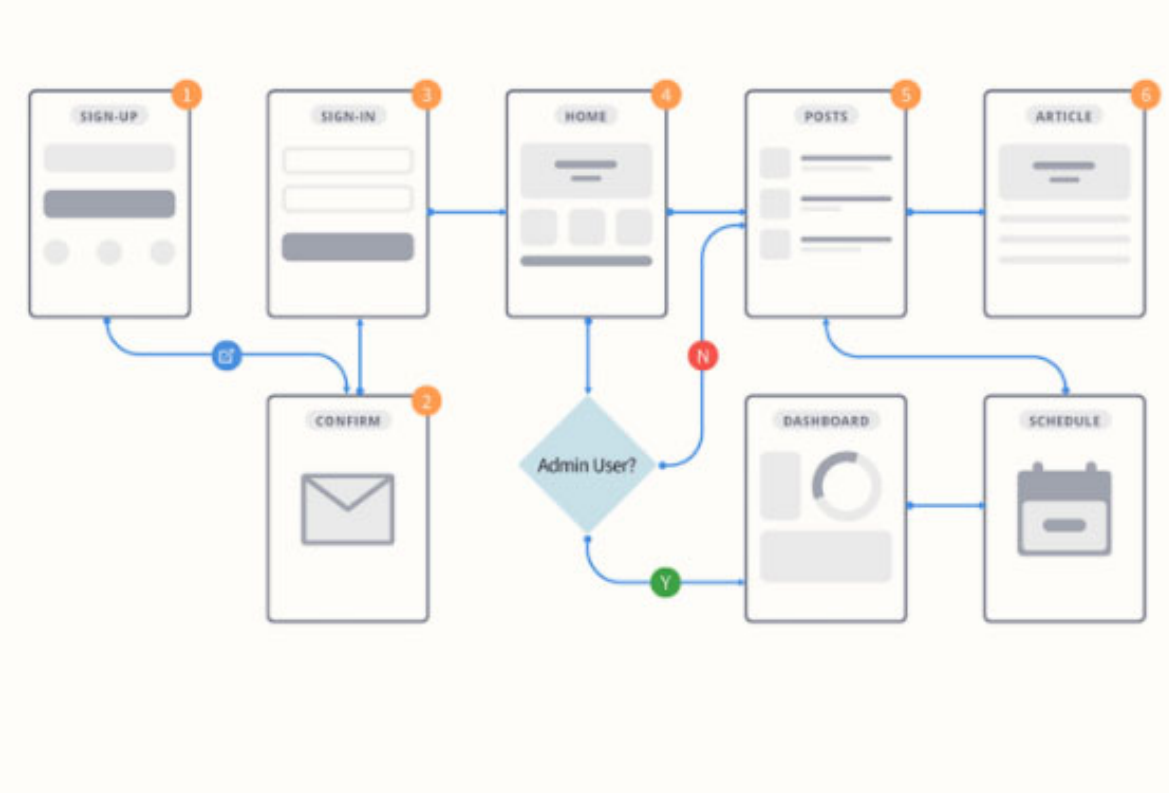
התרשים נלקח מהספר של פרץ שובל. (2004). *ניתוח ועיצוב מערכות מידע: שילוב תהליכים ועצמים* (Vol. 13021). Open University of Israel.

Interface tree diagram

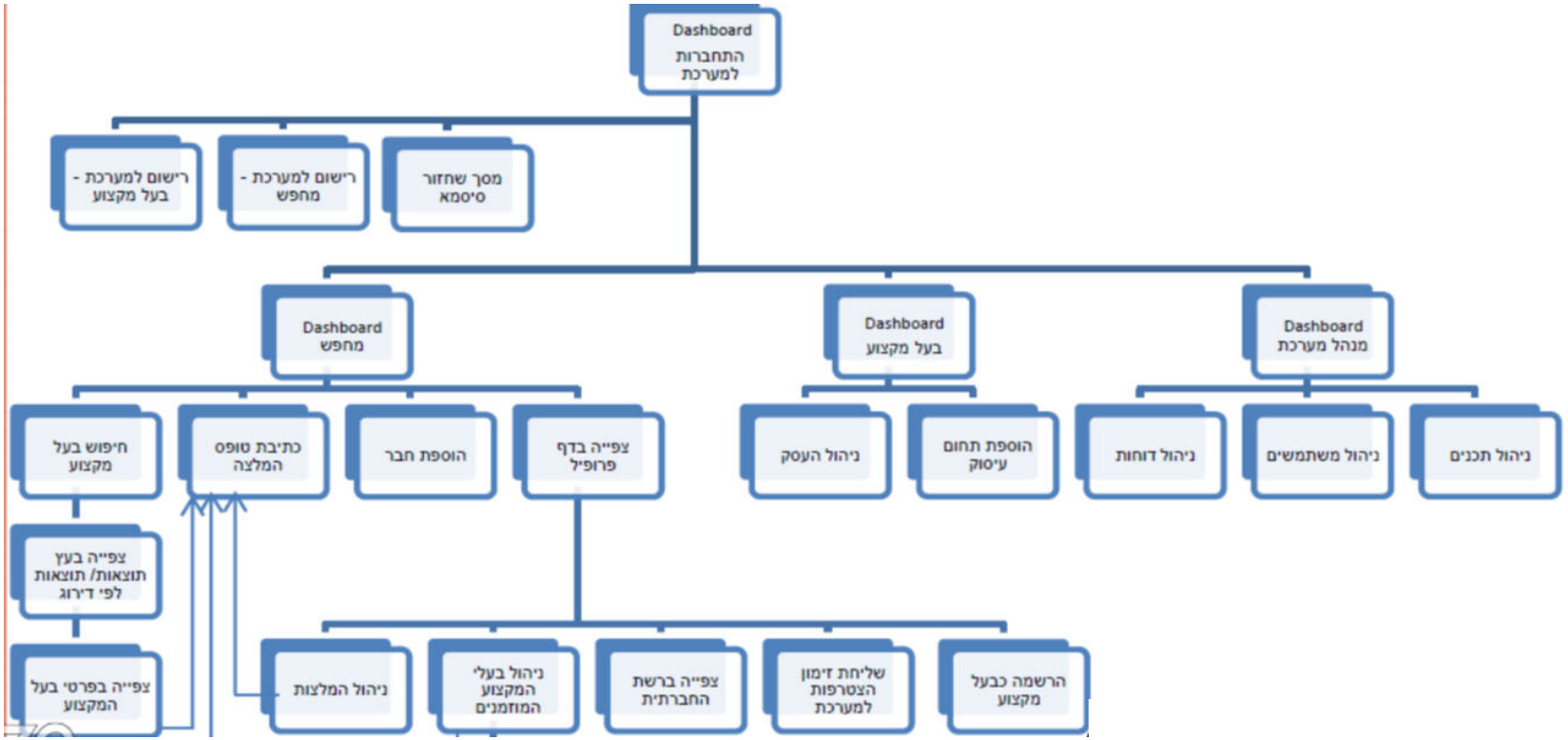


Lucidchart, Figma, invasion, visual paradigm (license for SCE students)

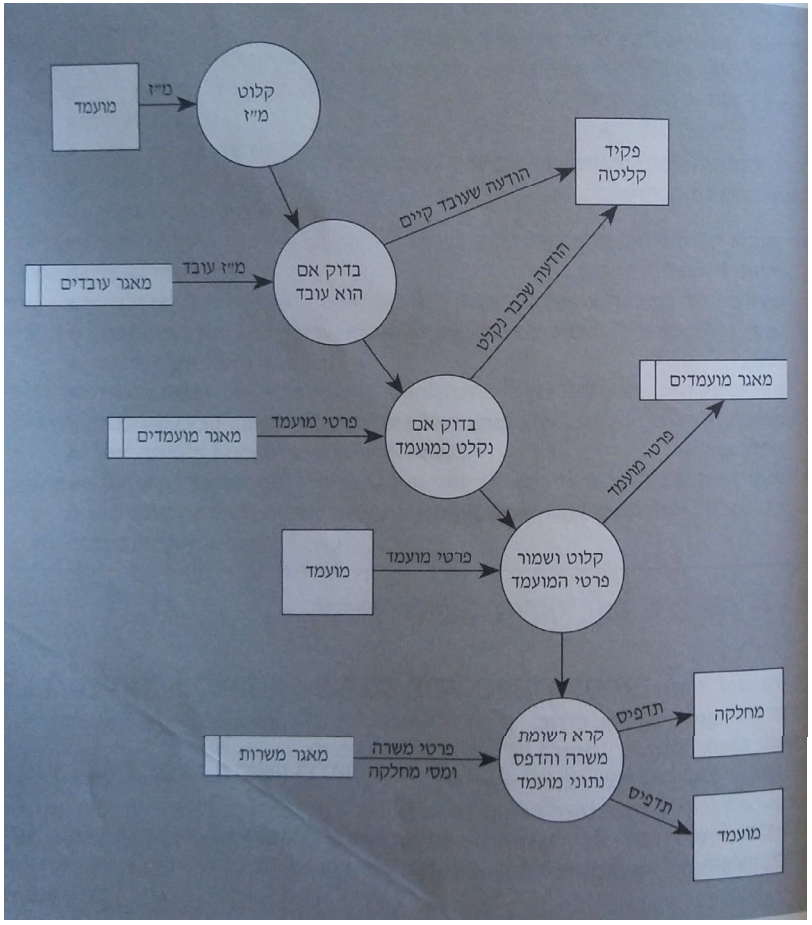
System/User (flow) Experience



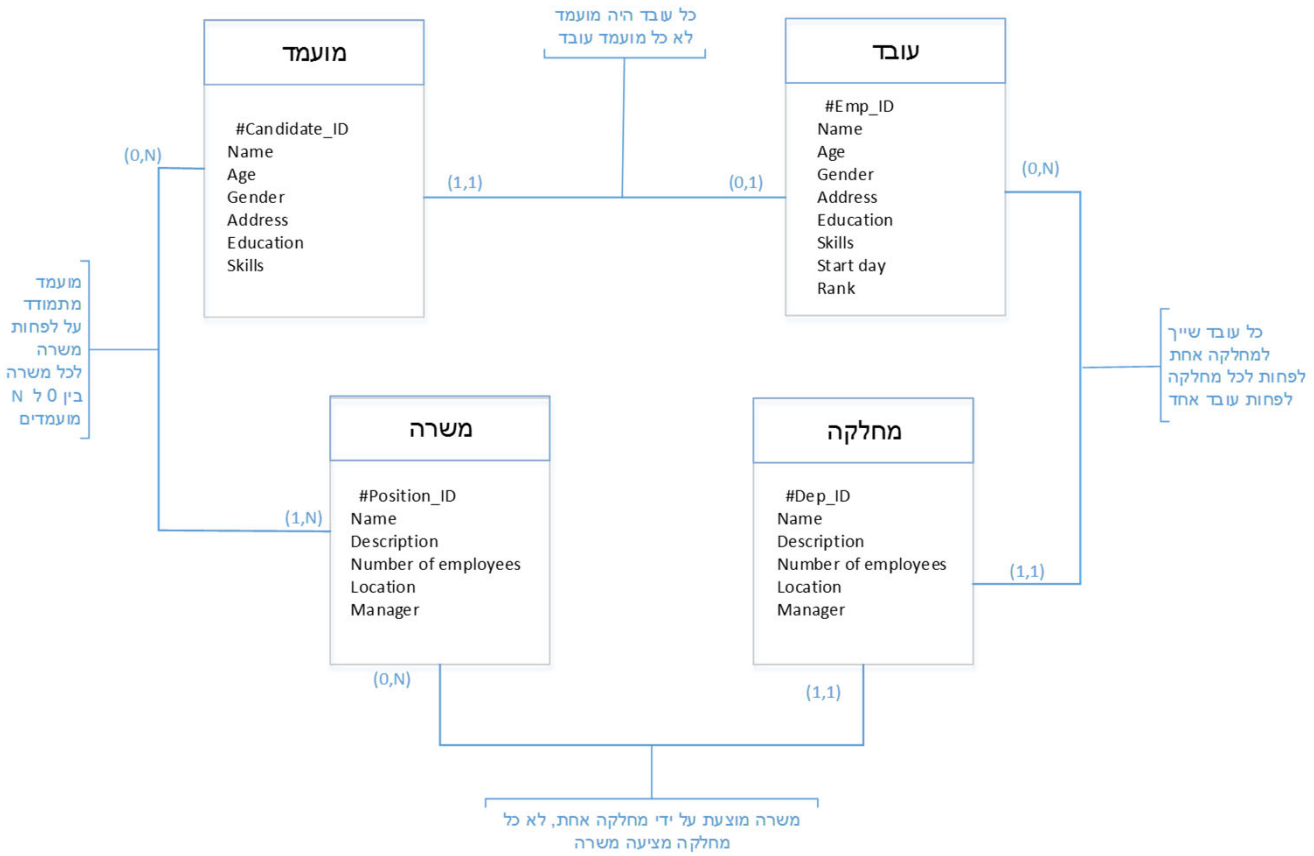
Interfaces flowchart



Data flow diagram (DFD)



Entity Relationships diagram



Architectural design

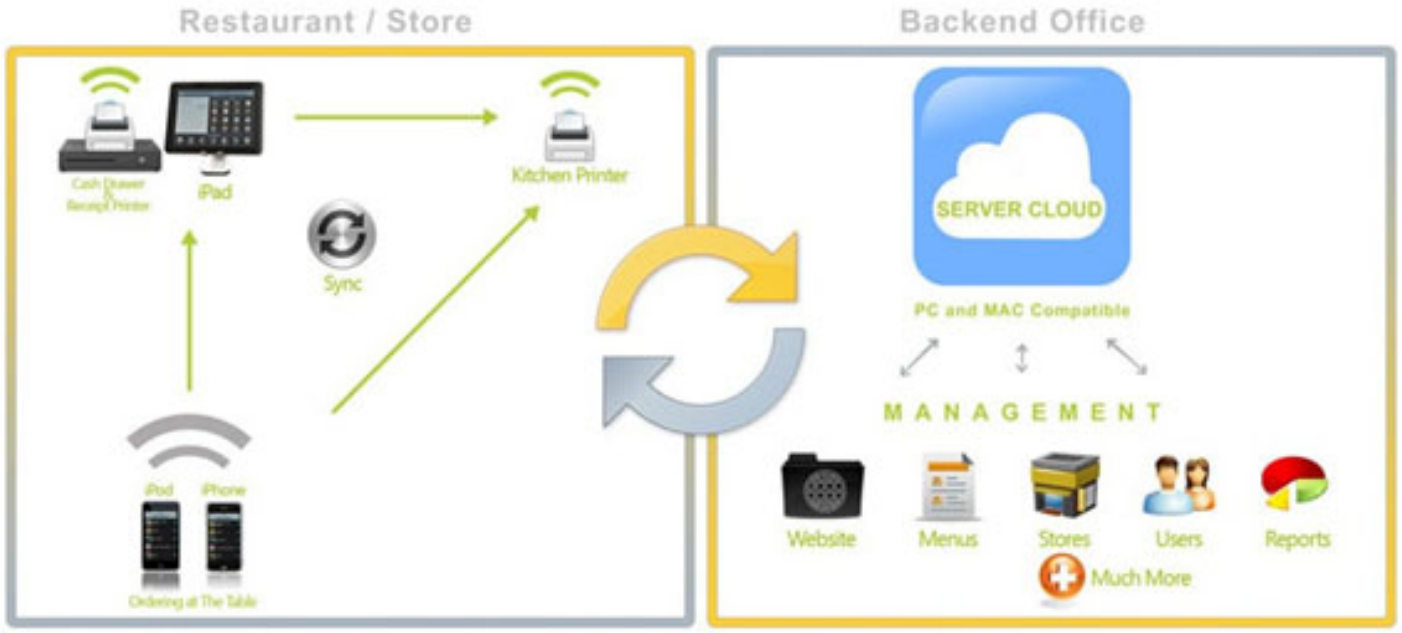
- How the system is organized, what is the overall structure
- Components and relations between them
- It is the first step of the design
- Design and documenting the software architecture contributes to:
  - *Stakeholder communication*
  - *System analysis* Making
  - *Large-scale reuse*

Block diagram

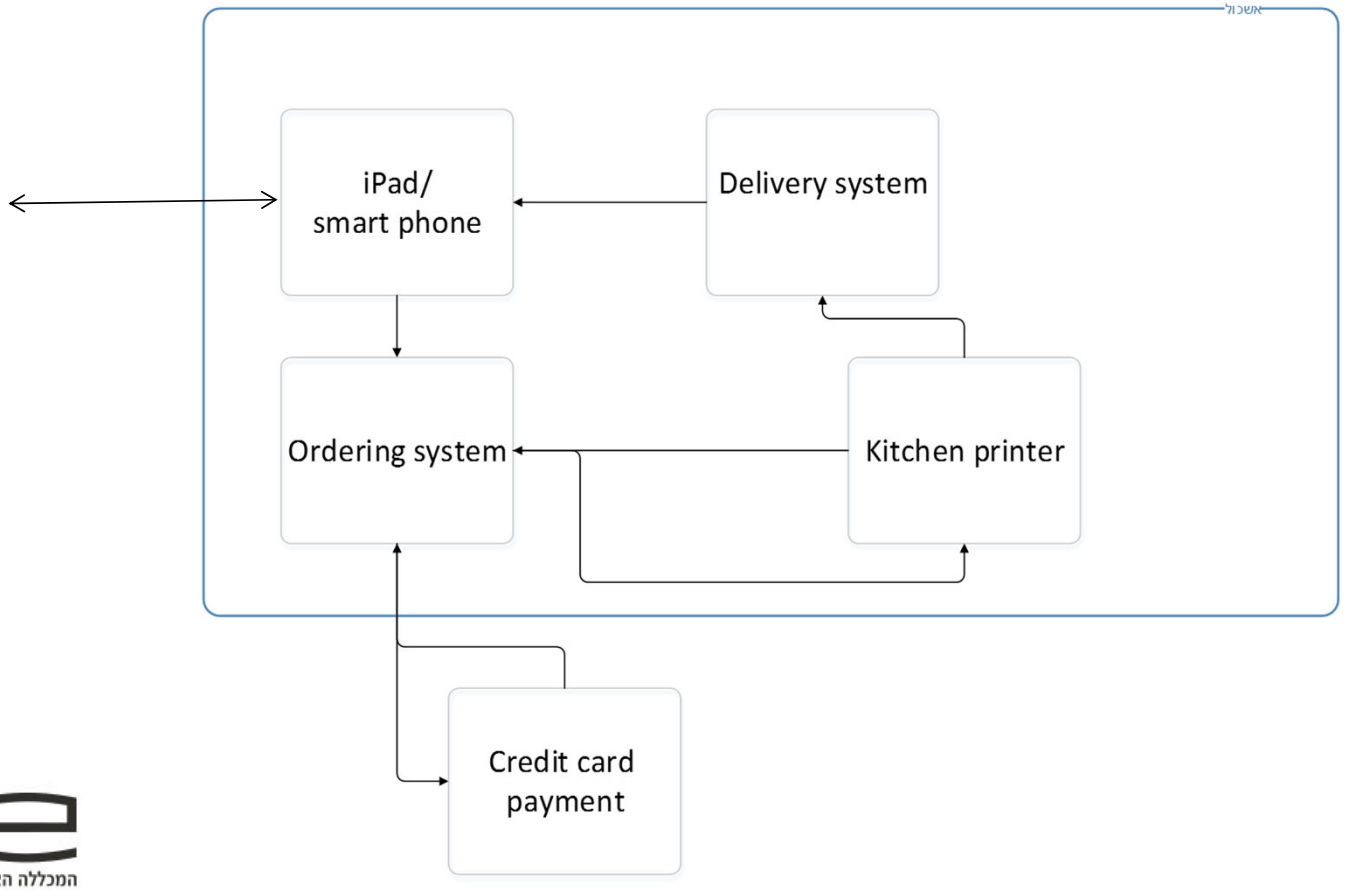
- High level picture of the system structure
- However, block diagrams don’t consider the type of the relationships among the components and their properties
  - A basis to discuss the system design
  - Can be used a way to document the architecture that has already been designed

Japanese ordering system

Online food ordering



Online food Ordering



Slide 1

Slide 2

Slide 3

Slide 4

Zoom in/out process

- Architecture in the small
- Architecture in the large

Gas station complex

Non-functional Requirements and Software Architecture

- **Performance** –localization of the processes within small number of components
- **Security**-dedicated layer with security validation
- **Safety**- the related operations should be located in either a single component or in a small number of components
- **Availability-** system should include redundant components (as a backup)
- **Maintainability-** using fine-grain to be easily changed

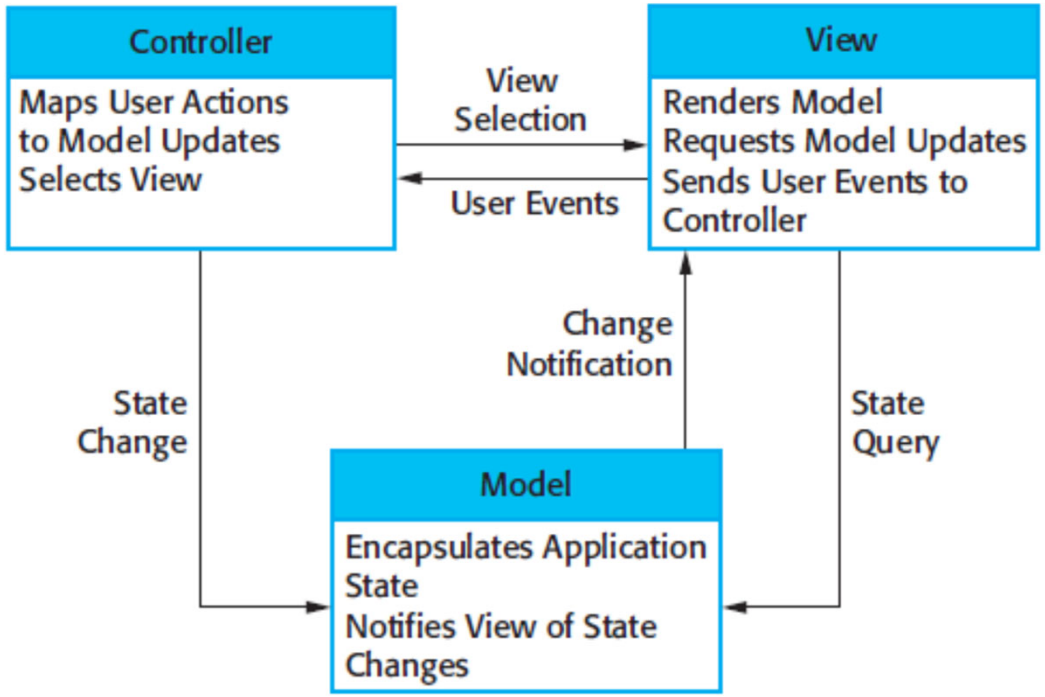
What is the problem?

4+1 view model of software architecture

1. Logical view
2. Process view
3. Development view
4. Physical view
5. + conceptual view

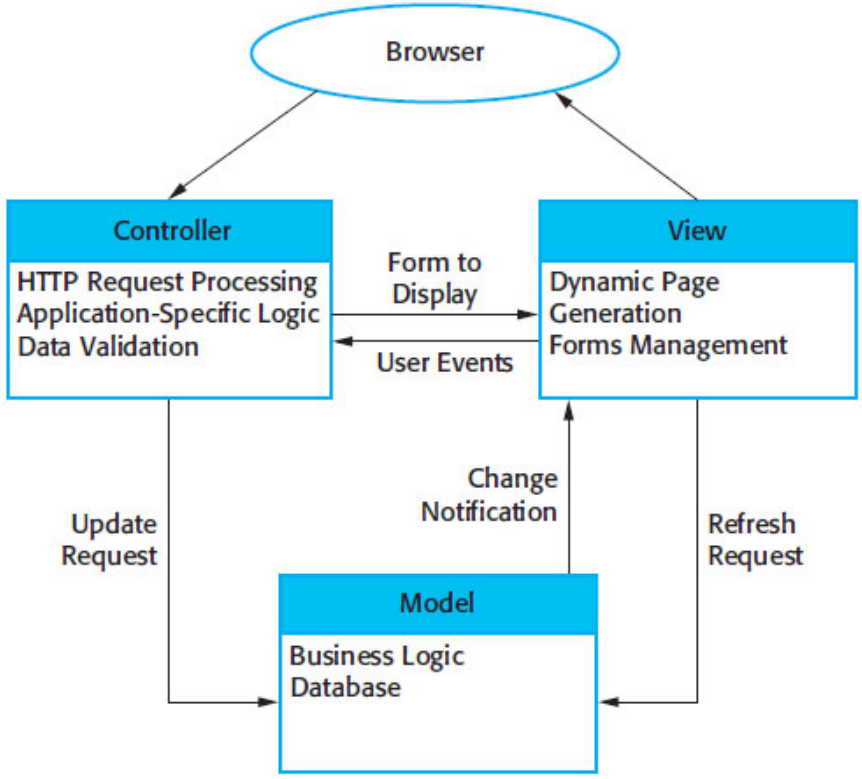
Tradeoff between detailed design and added value…

Model view controller (MVC)



"+" is the flexibility and in dependability between components "-" more code when it is a simple model

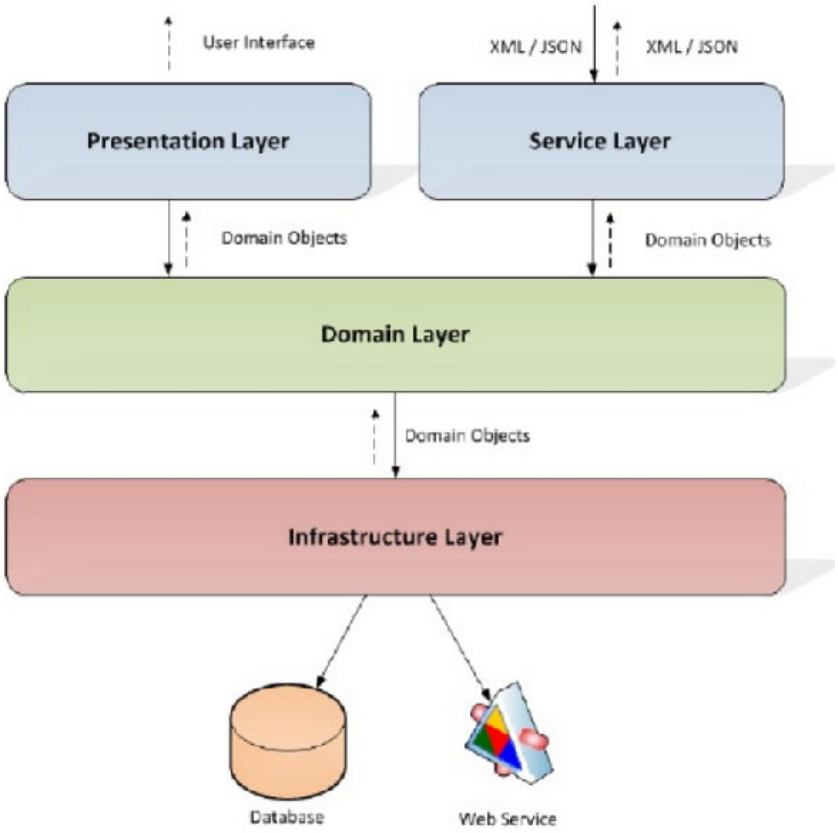
Web application-using mvc pattern

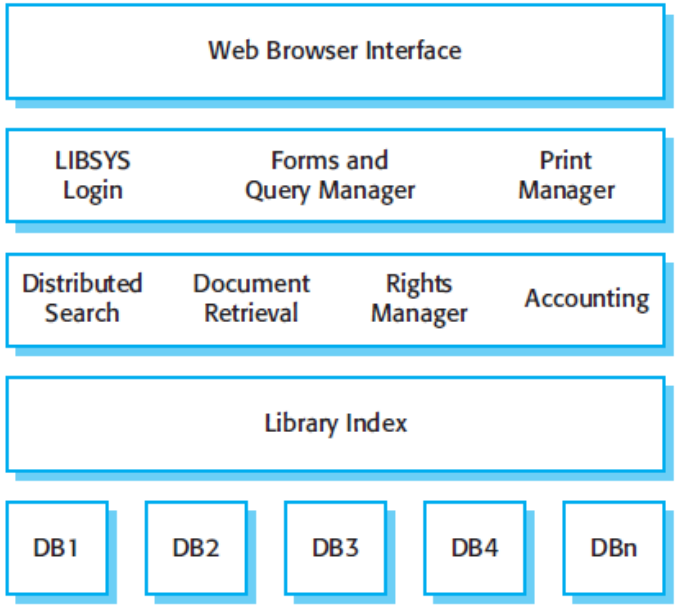


Layers approach

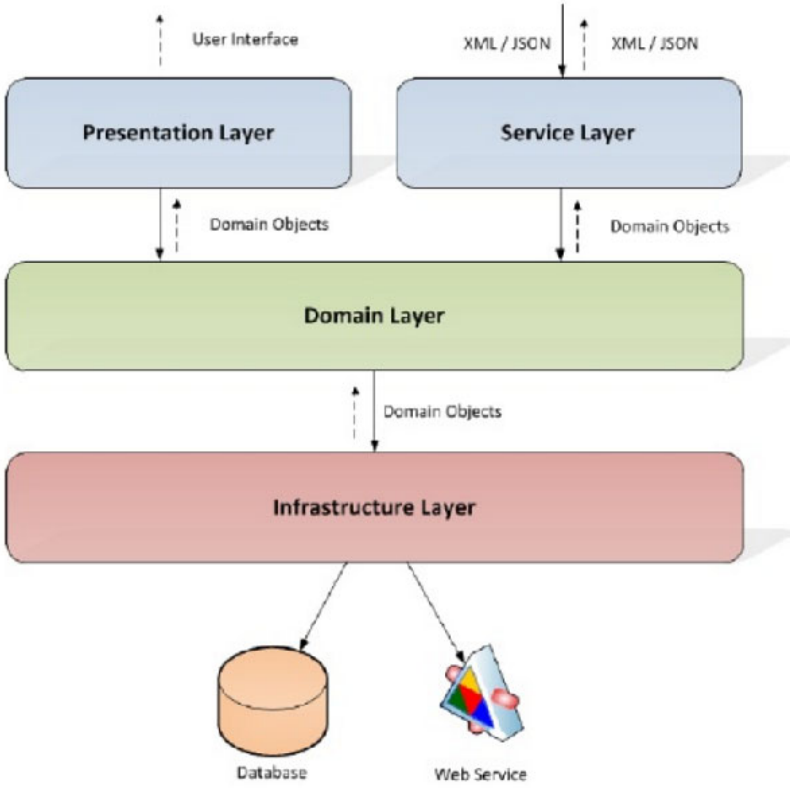
- Separation between the layers
- Each layer relies on the layer immediately beneath
- When adding new component/services to an existing system
- Distributed teams
- Difficult to implement “clean” interaction between layers
- The performance might be inferior due to the multiple levels of service request, which is processed at each level

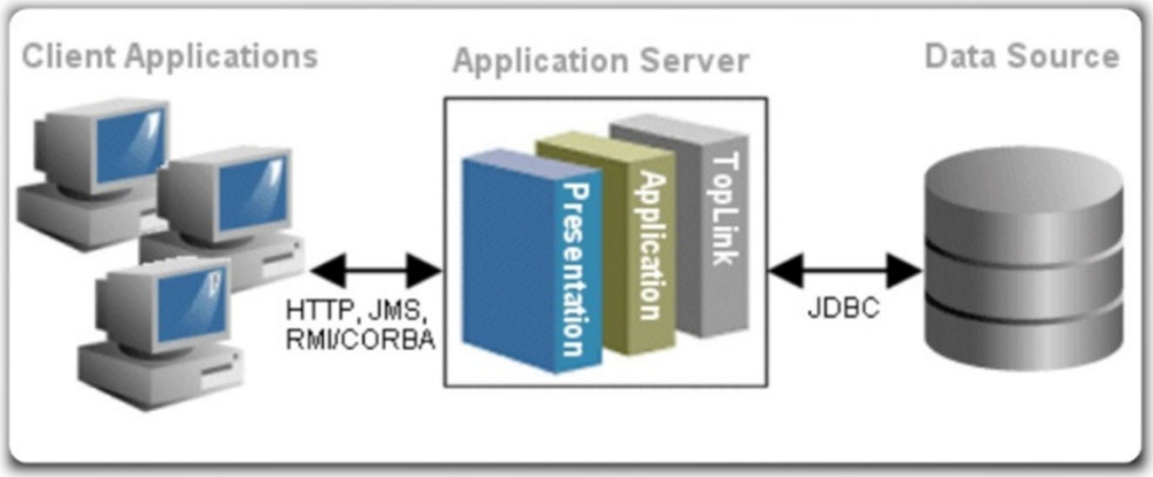
Generic layered architecture





3 tier architecture





3 tier Layers architecture- example

- Banking application with customers and accounts:
  - Account id, balance, and credit limit.
  - Over down account= is considered as “Sick” otherwise “healthy”.
- Use case I: user checks whether customer is sick/healthy.
- Use case II: A user wants to retrieve if a customer is healthy or sick.
- Technology .NET, .ASP MVC

- What do we need?
  1. Domain model for customer and account
  2. Business logic

1. 2 classes customer and list of accounts
2. Service class for building customer with accounts

3 tier layers Vs mvc

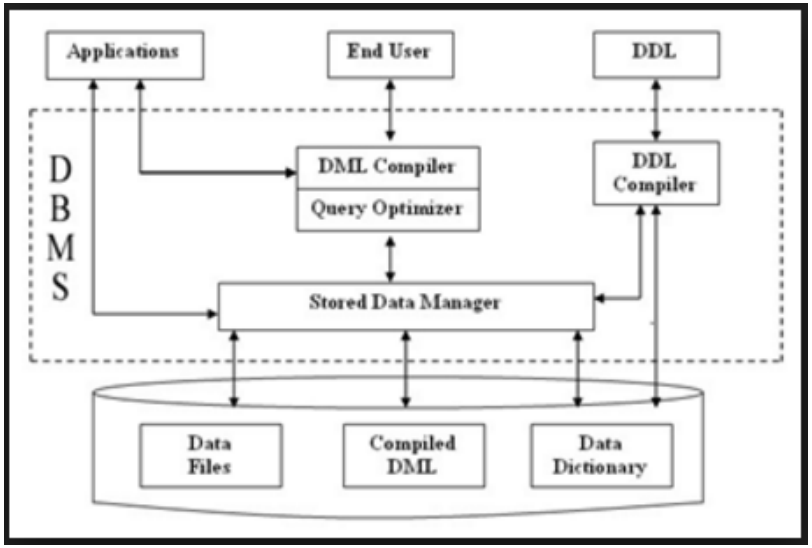
- Topologically is different
  - MVC is triangular
  - In the 3 layers client tier never communicates directly with the data tier
  - In the 3 layers all communication must pass through the middle tier
  - In MVC the view →controller → model →view

Logical layers vs physical tiers

- Logical layers
  - How you logically divide the code in the application
  - The most common layers: presentation/service layer; domain layer and model layer and domain services; infrastructure layer for communicating with the data sources
- Physical layer
  - How do you divide the application to many sub-applications

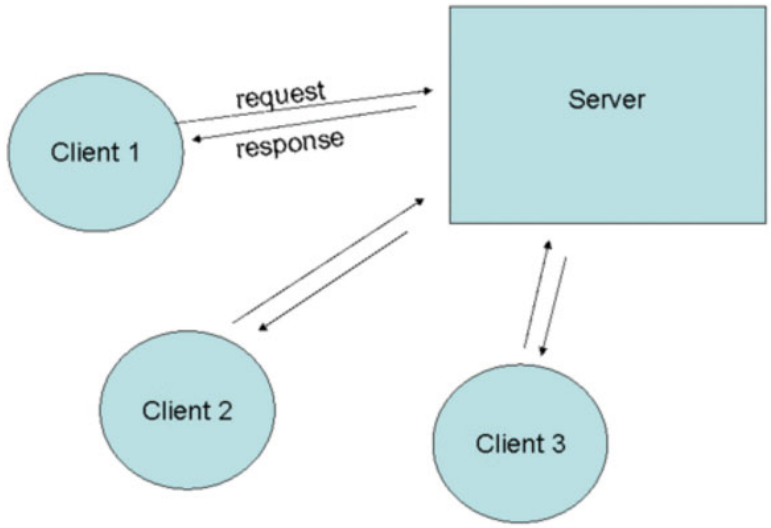
Repository architecture

- All data is managed in one central repository
- Objects communicate via the repository
- Is used in data-driven systems
- One point to manage Large amount of data
- Should fit all the components
- Failure in the repository affects all the related components



Client-server

- Composed of services (to the clients)
- Run-time architecture
- Servers can be distributed across a network and are owned by different organizations
- Each service is a single point of failure
- Performance might not be predictable

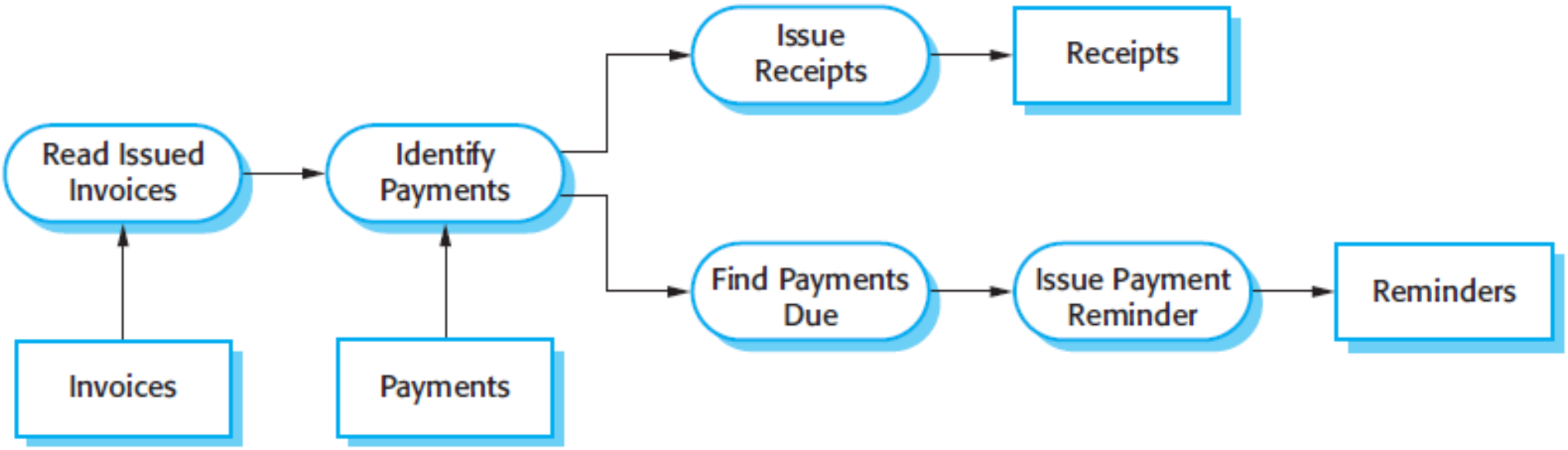


Pipe and filter architecture

- Run-time organization of a system where functional transformations

  process their inputs and produce outputs
- Suitable for batch applications (billing or embedded)
- Data can be processed sequentially or in parallel
- Originated in Unix systems when processes were linked by ‘pipes’.
- It is not appropriate for Interactive systems that process a stream of data at once.

Pipe and filter model



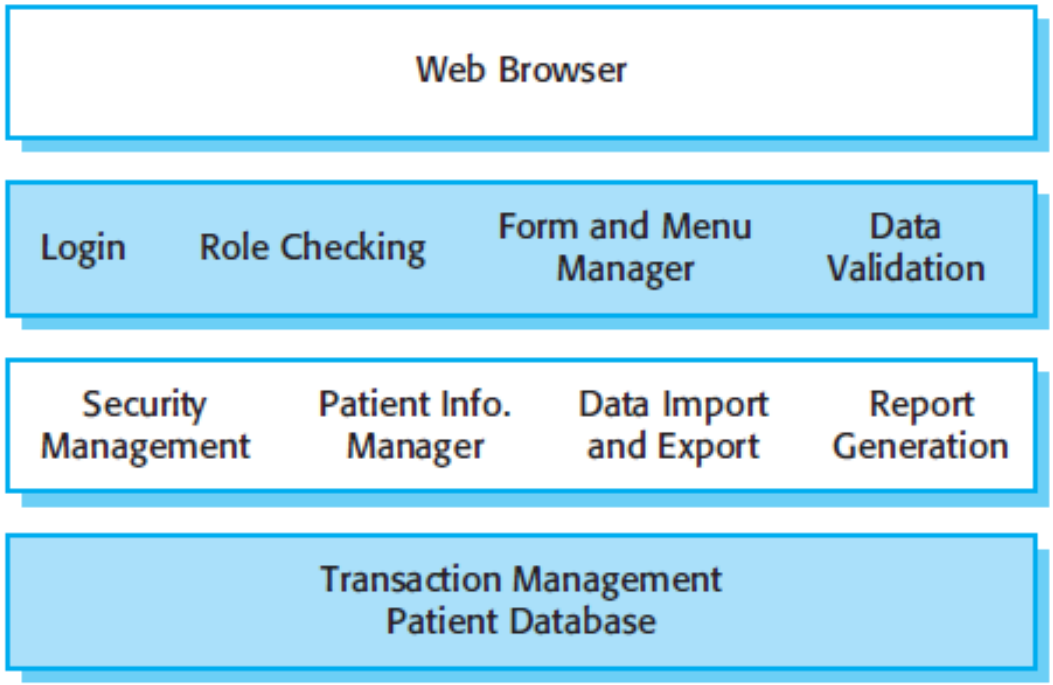
Application architecture

- Support the organizational strategy/business
- Tailored to a specific application, e.g., data collection systems or monitoring systems, ERP, COTS
- Can be reused, based on previous system implementation
- Can be implemented as:
  - *a starting point for the architectural design process*
  - *a design checklist*
  - *a way of organizing the work of the development team*
  - *means of assessing components for reuse*
  - *a vocabulary for talking about types of applications*

Information systems and transaction system

May be organized as a ‘pipe and filter’ architecture: input, processing, and output	Can be implement by using layer approach
Involve interaction with the data base	Allows controlled access to a large base of information

Information systems



Quiz -CHASSIDIM

Lets go to SOCRATIVE: https://b.socrative.com/login/student/
Room: CHASSIDIM

Take Home Messages

1. How to define good requirement
2. Interface design
3. Architecture concepts
4. Read the relevant chapters according to the syllabus
5. Prepare questions for the guidance meeting