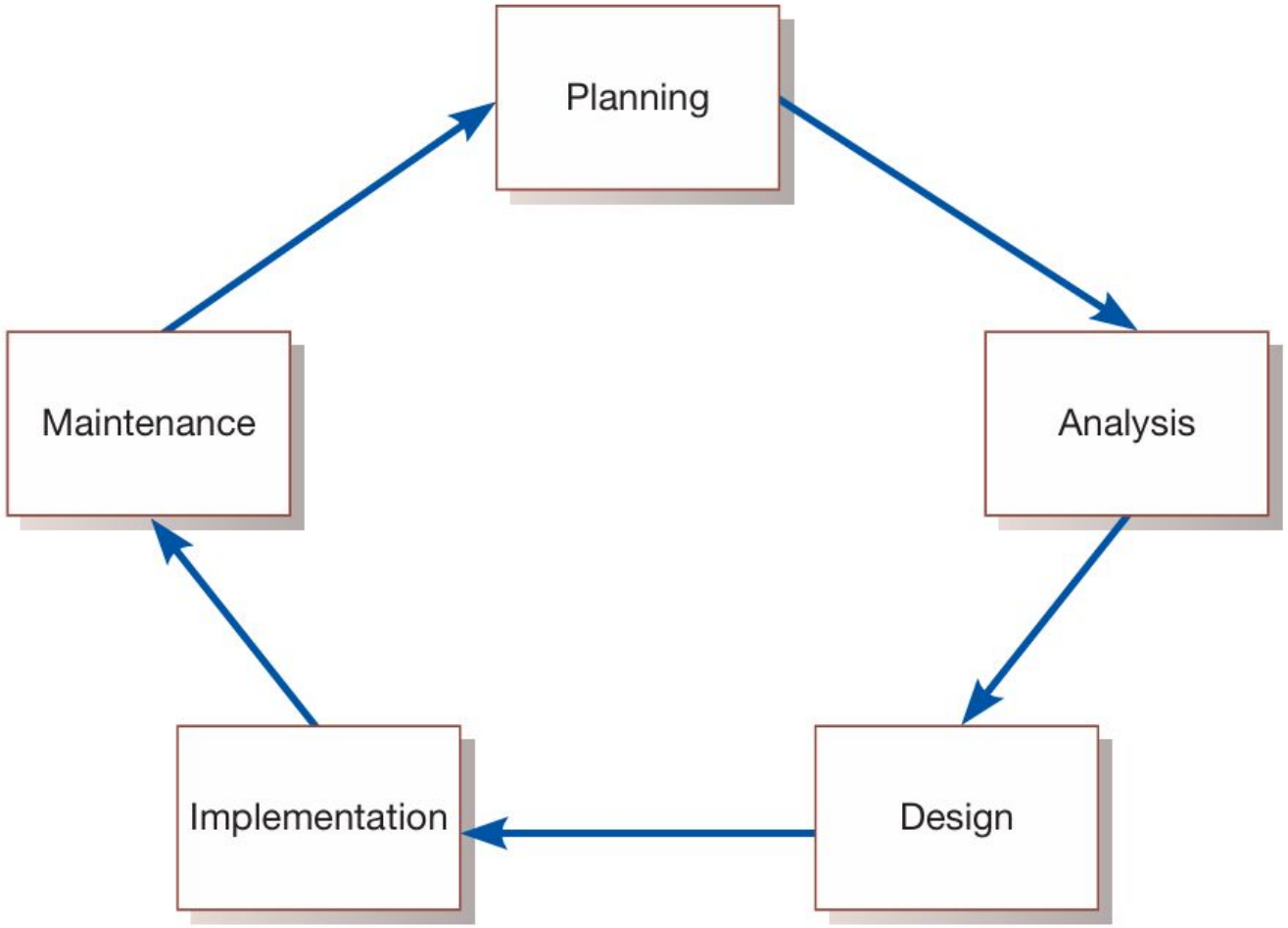
System Anaylsis and Design

Ibrahim Elsemman

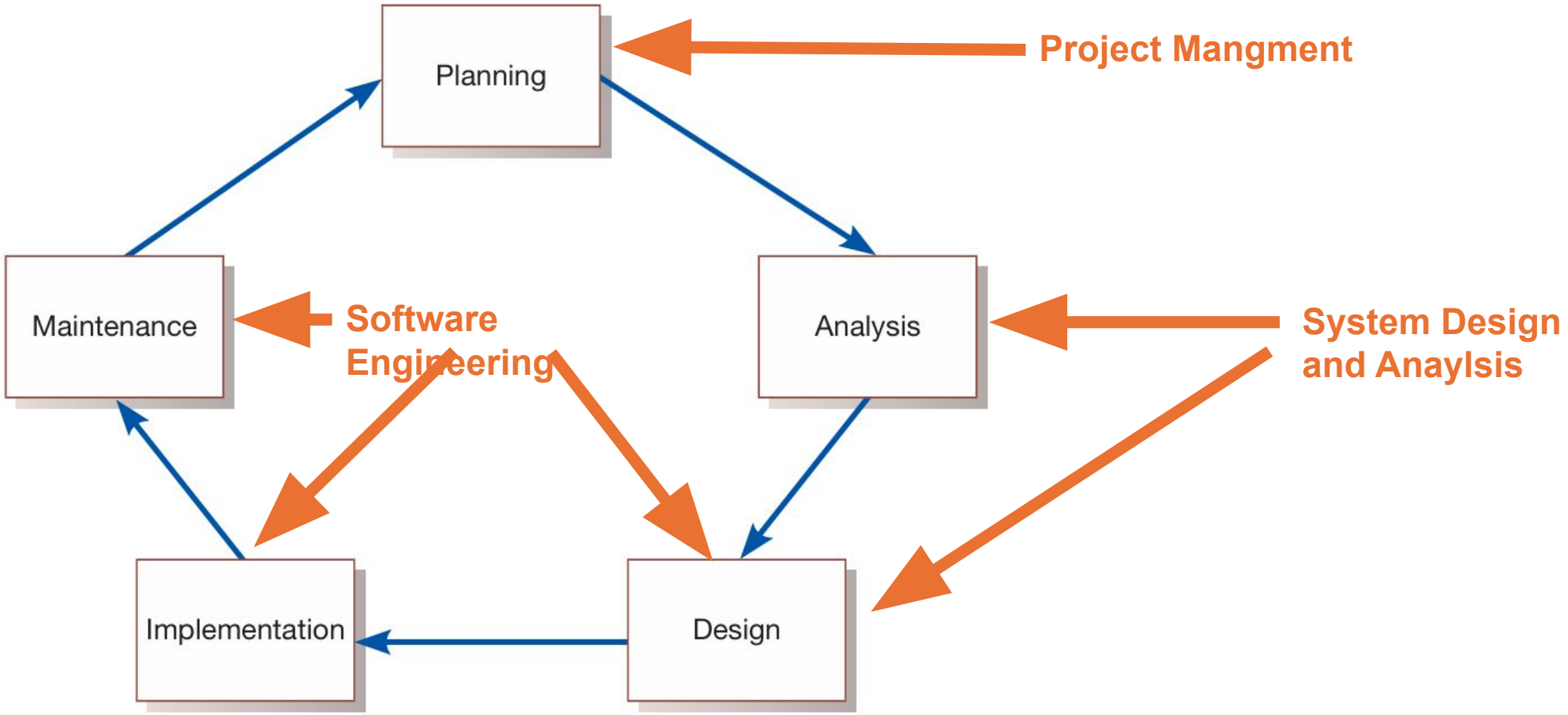
Four Important Courses to Make Enterprise software

- Project Mangment
- System Design and Anaylsis
- Software Engineering
- User Experience (UX)

Systems development life cycle



Systems development life cycle

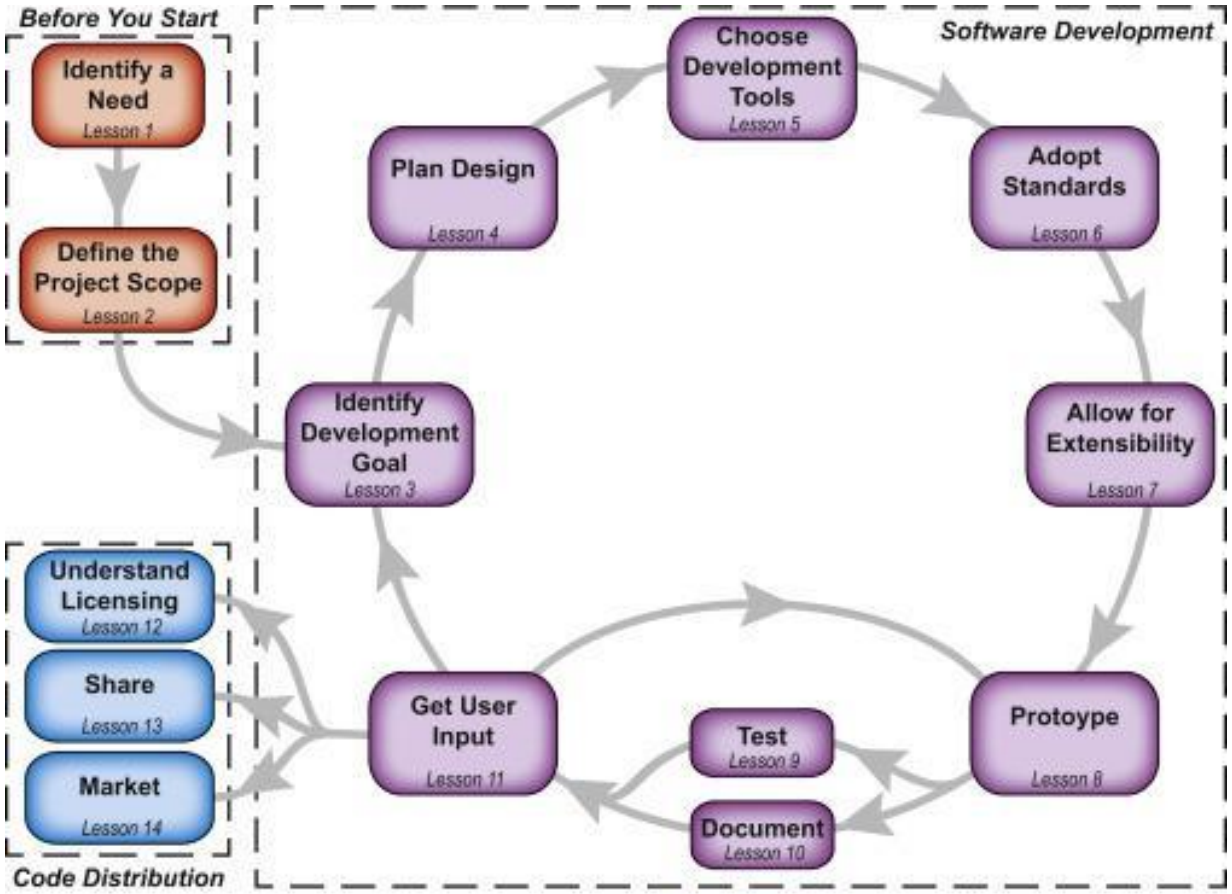


Do we really need project managment in software developlment?

- A survey of hundreds of corporate software development projects indicated that five out of six software projects are considered unsuccessful [Johnson 95], and approximately a third of software projects are canceled. The remaining projects delivered software at almost double the expected budget and time to develop as originally planned.

William J. Brown et al, AntiPatterns Refactoring Software, Architectures, and Projects in Crisis

Software Development Process



Visual Description of How the 14 steps during the Software Development Process

What Is a Project?

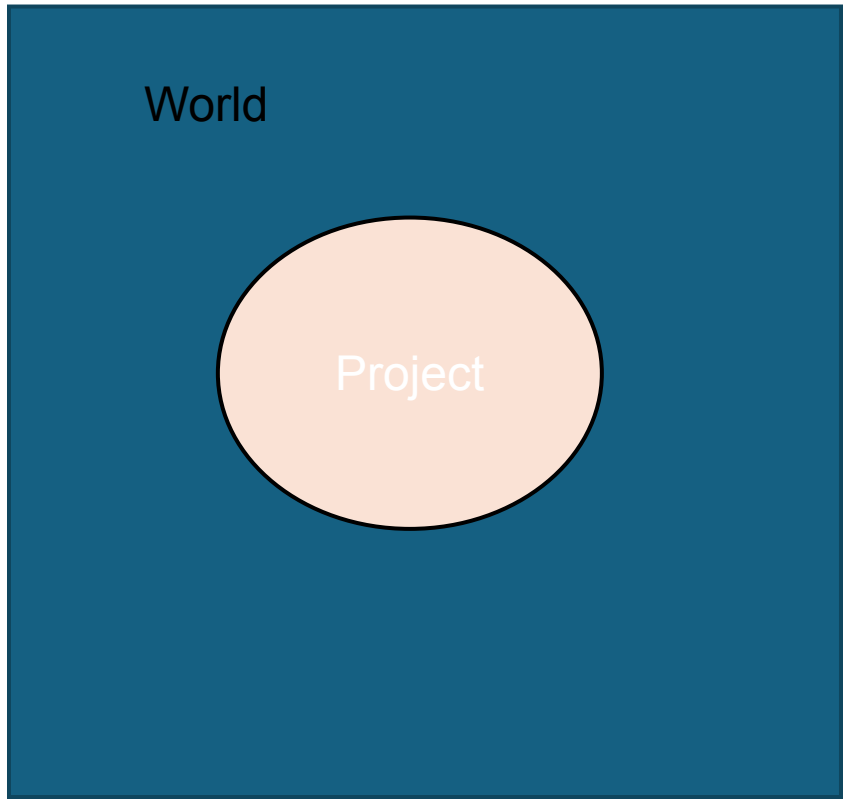
- *A **project** is a temporary endeavor undertaken to create a **unique product, service, or result**.*

Defining the Project Scope

- You should know that:
- My project do these, but not do these

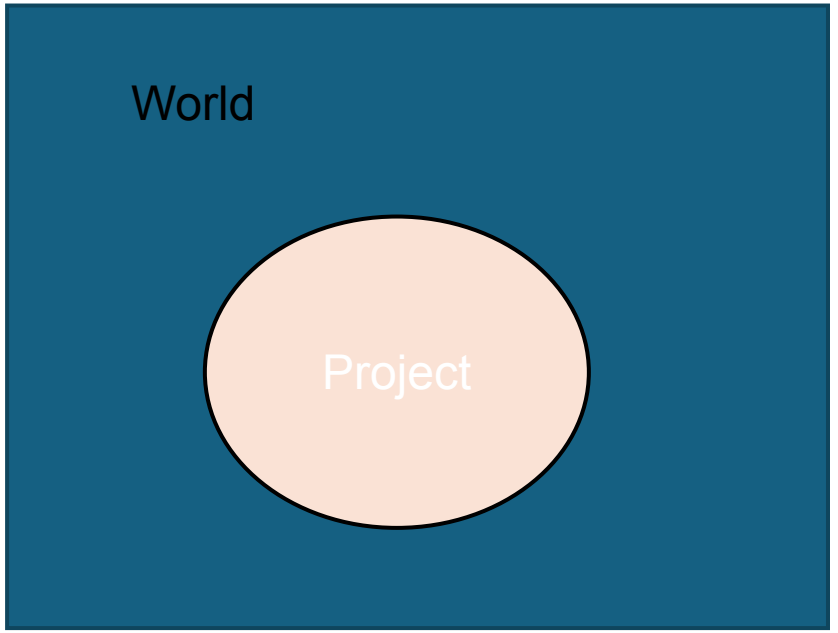
Project Scope Checklist

1. Project objective
2. Deliverables
3. Milestones
4. Technical requirements
5. Limits and exclusions
6. Reviews with customer



Defining the Project Scope

- You should know that:
- My project do these, but not do these



Project Scope Checklist

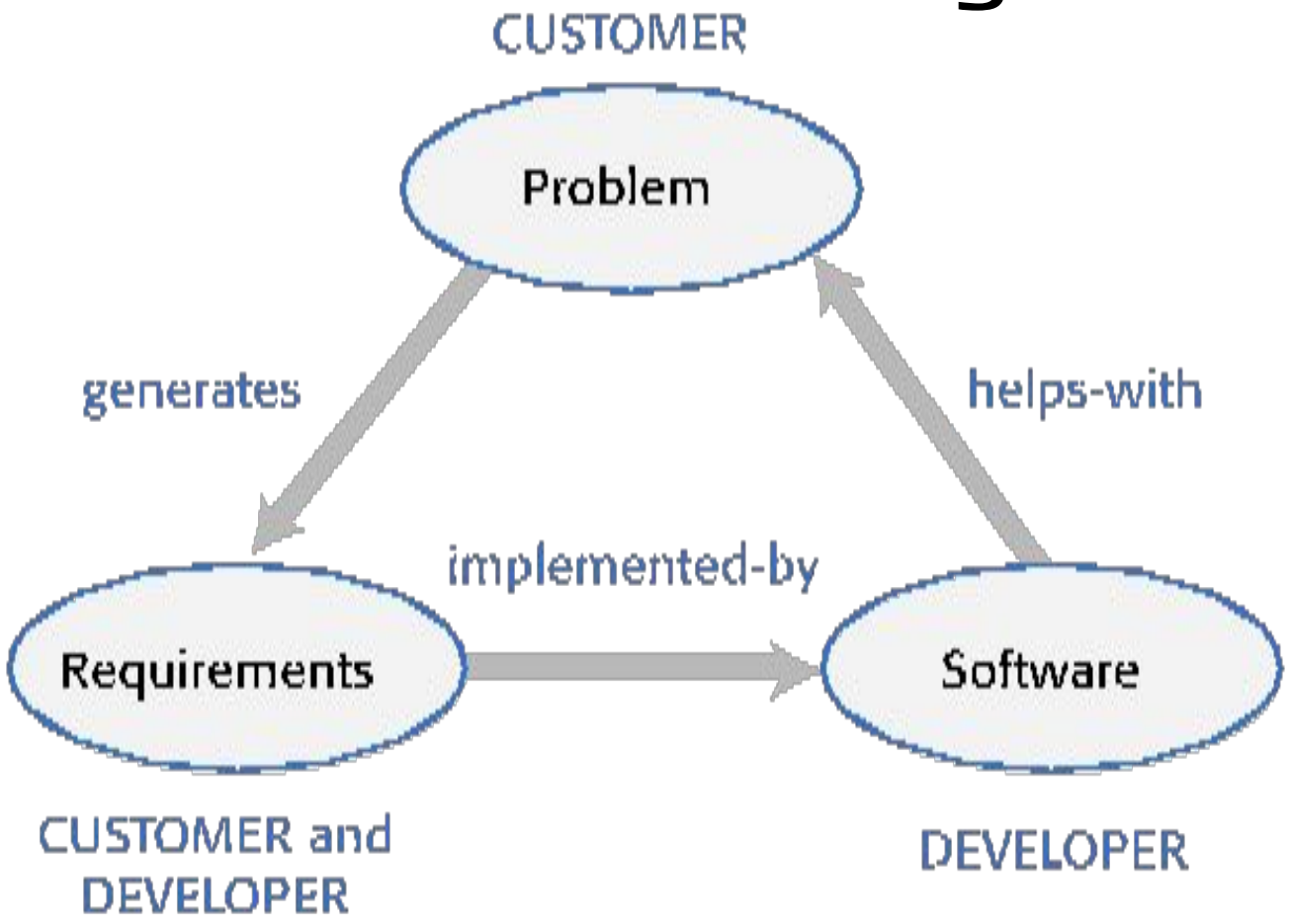
1. Project objective
2. Deliverables
3. Milestones
4. Technical requirements
5. Limits and exclusions
6. Reviews with customer

First Step in Software Engineering

Software products

- Software products are generic software systems that provide functionality that is useful to a range of customers.
- Many different types of products are available from large-scale business systems (e.g. MS Excel) through personal products (e.g. Evernote) to simple mobile phone apps and games (e.g. Suduko).
- Software product engineering methods and techniques have evolved from software engineering techniques that support the development of one-off, custom software systems.
- Custom software systems are still important for large businesses, government and public bodies. They are developed in dedicated software projects.

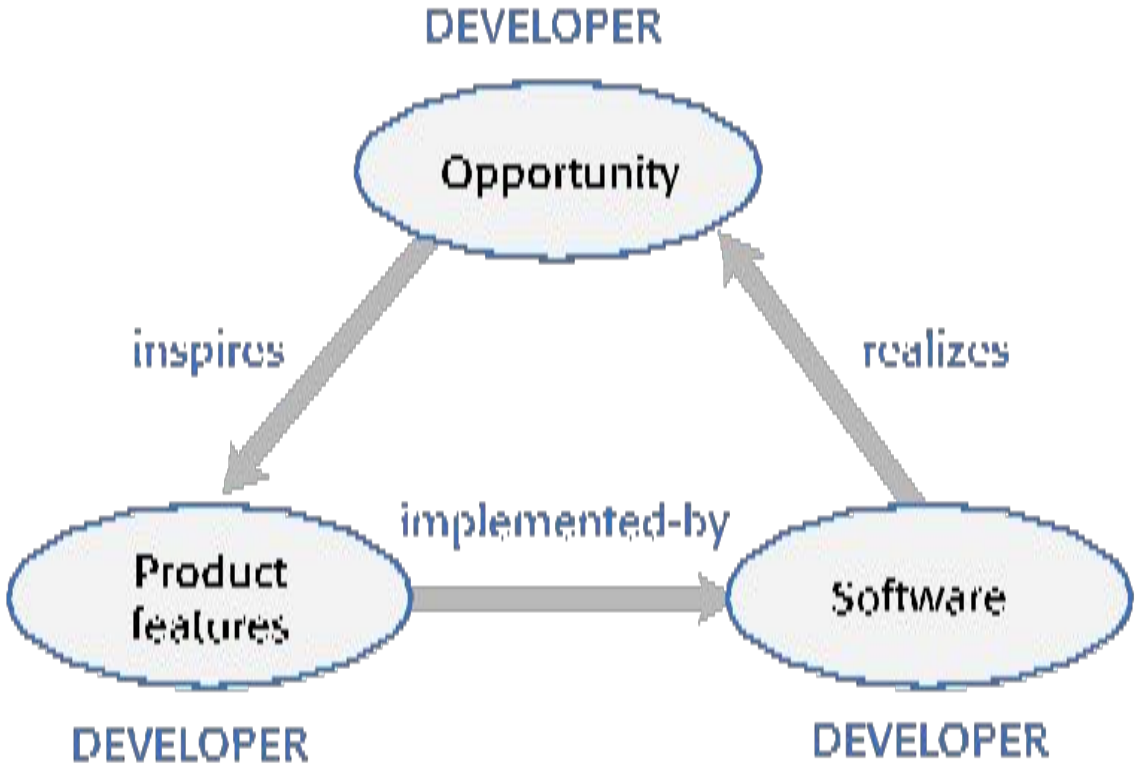
Project-based software engineering



Project-based software engineering

- The starting point for the software development is a set of ‘software requirements’ that are owned by an external client and which set out what they want a software system to do to support their business processes.
- The software is developed by a software company (the contractor) who design and implement a system that delivers functionality to meet the requirements.
- The customer may change the requirements at any time in response to business changes (they usually do). The contractor must change the software to reflect these requirements changes.
- Custom software usually has a long-lifetime (10 years or more) and it must be supported over that lifetime.

Product software engineering



Product software engineering

- The starting point for product development is a business opportunity that is identified by individuals or a company. They develop a software product to take advantage of this opportunity and sell this to customers.
- The company who identified the opportunity design and implement a set of software features that realize the opportunity and that will be useful to customers.
- The software development company are responsible for deciding on the development timescale, what features to include and when the product should change.
- Rapid delivery of software products is essential to capture the market for that type of product.

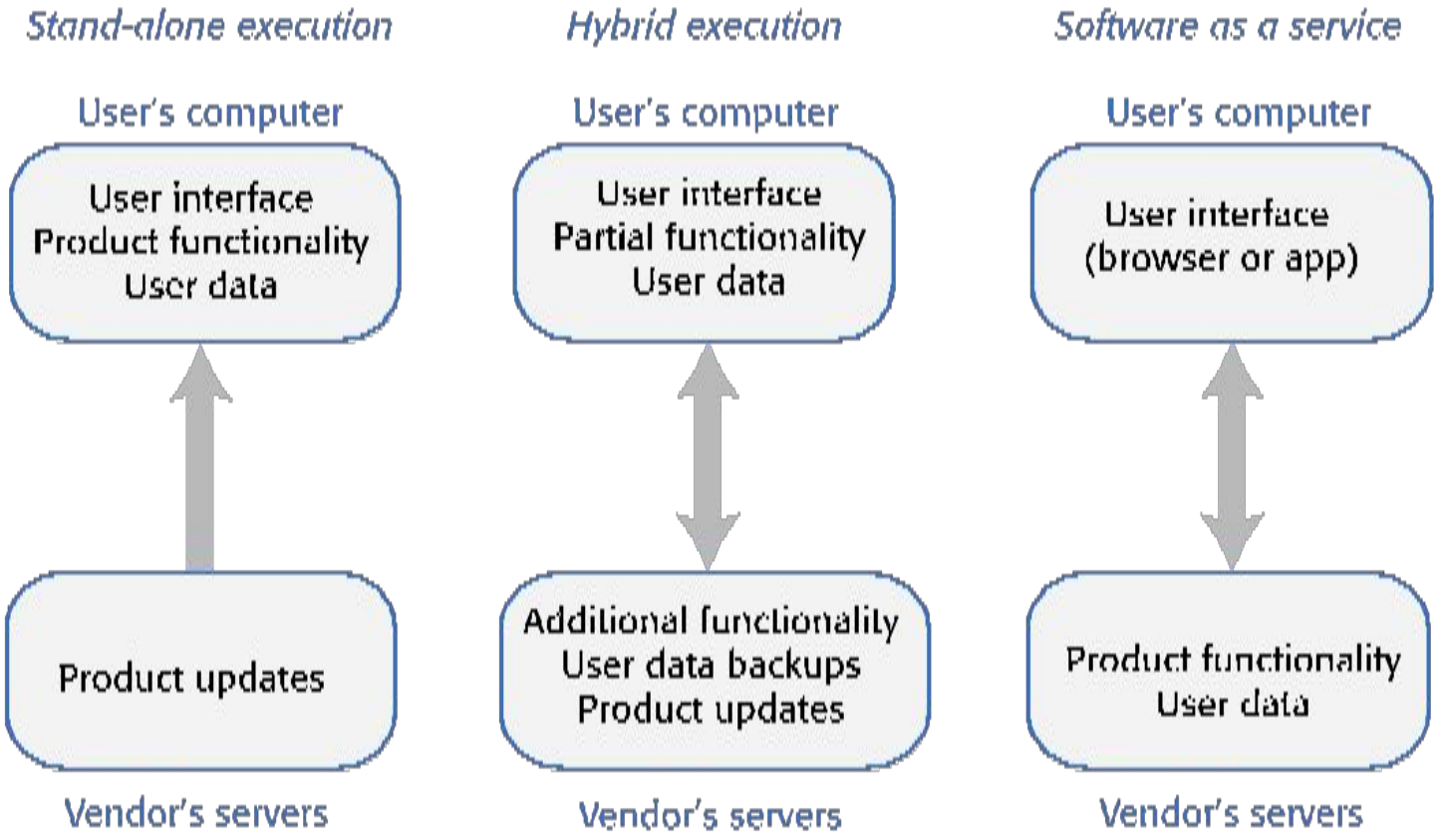
Software product lines and platforms

- ***Software product line***
  A set of software products that share a common core. Each member of the product line includes customer-specific adaptations and additions. Software product lines may be used to implement a custom system for a customer with specific needs that can't be met by a generic product.
- ***Platform***
  A software (or software+hardware) product that includes functionality so that new applications can be built on it. An example of a platform that you probably use is Facebook. It provides an extensive set of product functionality but also provides support for creating 'Facebook apps'. These add new features that may be used by a business or a Facebook interest group.

Software execution models

- ***Stand-alone*** The software executes entirely on the customer's computers.
- ***Hybrid*** Part of the software's functionality is implemented on the customer's computer but some features are implemented on the product developer's servers.
- ***Software service*** All of the product's features are implemented on the developer's servers and the customer accesses these through a browser or a mobile app.

Software execution models



Comparable software development

- The key feature of product development is that there is no external customer that generates requirements and pays for the software. This is also true for other types of software development:
  - ***Student projects*** Individuals or student groups develop software as part of their course. Given an assignment, they decide what features to include in the software.
  - ***Research software*** Researchers develop software to help them answer questions that are relevant to their research.
  - ***Internal tool development*** Software developers may develop tools to support their work - in essence, these are internal products that are not intended for customer release.

The product vision

- The starting point for software product development is a ‘product vision’.
- Product visions are simple statements that define the essence of the product to be developed.
- The product vision should answer three fundamental questions:
  - What is the product to be developed?
  - Who are the target customers and users?
  - Why should customers buy this product?

Moore's vision template

- FOR (target customer)
- WHO (statement of the need or opportunity)
- The (PRODUCT NAME) is a (product category)
- THAT (key benefit, compelling reason to buy)
- UNLIKE (primary competitive alternative)
- OUR PRODUCT (statement of primary differentiation)

Vision template example

“FOR a mid-sized company's marketing and sales departments WHO need basic CRM functionality, THE CRM-Innovator is a Web-based service THAT provides sales tracking, lead generation, and sales representative support features that improve customer relationships at critical touch points. UNLIKE other services or package software products, OUR product provides very capable services at a moderate cost.”

Table 1.2 Information sources for developing a product vision

- ***Domain experience***
  The product developers may work in a particular area (say marketing and sales) and understand the software support that they need. They may be frustrated by the deficiencies in the software they use and see opportunities for an improved system.

- ***Product experience***
  Users of existing software (such as word processing software) may see simpler and better ways of providing comparable functionality and propose a new system that implements this. New products can take advantage of recent technological developments such as speech interfaces.

- ***Customer experience***
  The software developers may have extensive discussions with prospective customers of the product to understand the problems that they face, constraints, such as interoperability, that limit their flexibility to buy new software, and the critical attributes of the software that they need.

- ***Prototyping and playing around***
  Developers may have an idea for software but need to develop a better understanding of that idea and what might be involved in developing it into a product. They may develop a prototype system as an experiment and ‘play around’ with ideas and variations using that prototype system as a platform.

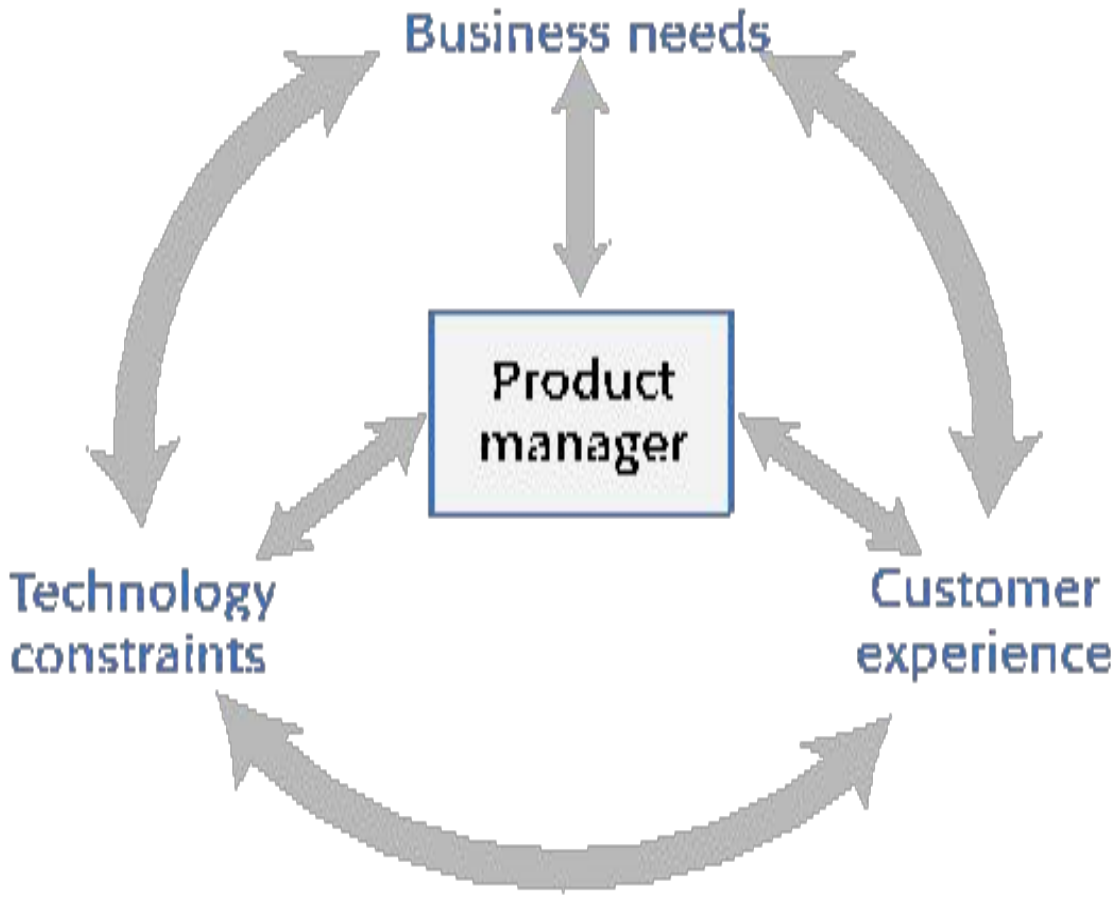
Table 1.3 A vision statement for the iLearn system

- **FOR** teachers and educators **WHO** need a way to help students use web-based learning resources and applications, **THE** iLearn system is an open learning environment **THAT** allows the set of resources used by classes and students to be easily configured for these students and classes by teachers themselves. **UNLIKE** Virtual Learning Environments, such as Moodle, the focus of iLearn is the learning process rather than the administration and management of materials, assessments and coursework. **OUR** product enables teachers to create subject and age-specific environments for their students using any web-based resources, such as videos, simulations and written materials that are appropriate.
- Schools and universities are the target customers for the iLearn system as it will significantly improve the learning experience of students at relatively low cost. It will collect and process learner analytics that will reduce the costs of progress tracking and reporting.

Software product management

- Software product management is a business activity that focuses on the software products developed and sold by the business.
- Product managers (PMs) take overall responsibility for the product and are involved in planning, development and product marketing.
- Product managers are the interface between the organization, its customers and the software development team. They are involved at all stages of a product's lifetime from initial conception through to withdrawal of the product from the market.
- Product managers must look outward to customers and

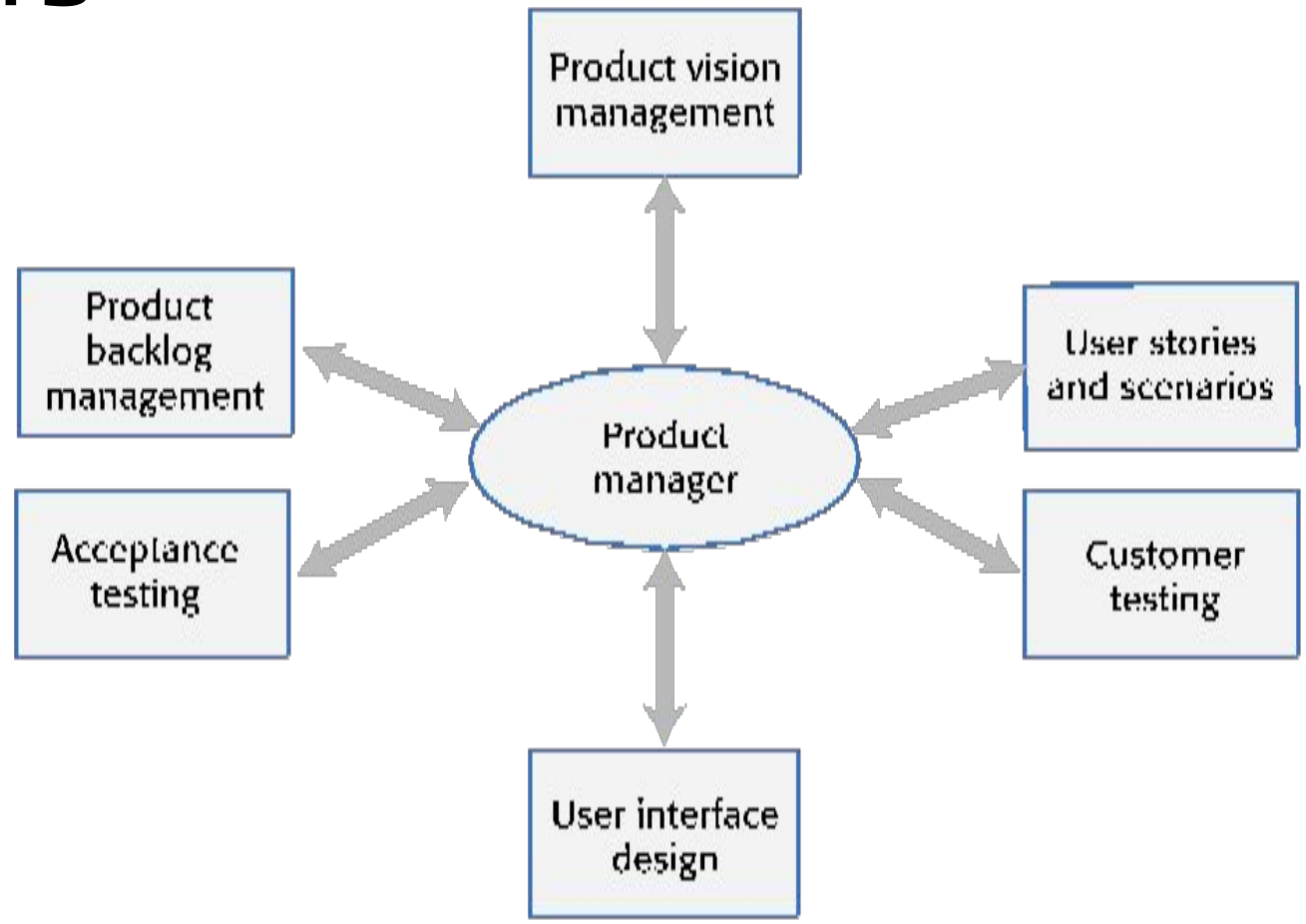
Product management concerns



Product management concerns

- ***Business needs*** PMs have to ensure that the software being developed meets the business goals of the software development company.
- ***Technology constraints*** PMs must make developers aware of technology issues that are important to customers.
- ***Customer experience*** PMs should be in regular contact with customers and potential customers to understand what they are looking for in a product, the types of users and their backgrounds and the ways that the product may be used.

Technical interactions of product managers



Technical interactions of product managers

- Product vision management
  - The product manager may be responsible for helping with the development of the product vision. The should always be responsible for managing the vision, which involves assessing and evaluating proposed changes against the product vision. They should ensure that there is no ‘vision drift’
- Product roadmap development
  - A product roadmap is a plan for the development, release and marketing of the software. The PM should lead roadmap development and should be the ultimate authority in deciding if changes to the roadmap should be made.
- User story and scenario development
  - User stories and scenarios are used to refine a product vision and identify product features. Based on his or her knowledge of customers, the PM should lead the development of stories and scenarios.

Technical interactions of product managers

- Product backlog creation and management
  - The product backlog is a prioritized ‘to-do’ list of what has to be developed. PMs should be involved in creating and refining the backlog and deciding on the priority of product features to be developed.
- Acceptance testing
  - Acceptance testing is the process of verifying that a software release meets the goals set out in the product roadmap and that the product is efficient and reliable. The PM should be involved in developing tests of the product features that reflect how customers use the product.
- Customer testing
  - Customer testing involves taking a release of a product to customers and getting feedback on the product’s features, usability and business. PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.
- User interface design
  - Product managers should understand user limitations and act as surrogate users in their interactions with the development team. They should evaluate user interface features as they are developed to check that these features are not unnecessarily complex or force users to work in an unnatural way.

Product prototyping

- Product prototyping is the process of developing an early version of a product to test your ideas and to convince yourself and company funders that your product has real market potential.
  - You may be able to write an inspiring product vision, but your potential users can only really relate to your product when they see a working version of your software. They can point out what they like and don't like about it and make suggestions for new features.
  - A prototype may be also used to help identify fundamental software components or services and to test technology.
- Building a prototype should be the first thing that you do when developing a software product. Your aim should be to have a working version of your software that can be used to demonstrate its key features.
- You should always plan to throw-away the prototype after development and to re-implement the software, taking account of issues such as security and reliability.

Two-stage prototyping

- ***Feasibility demonstration*** You create an executable system that demonstrates the new ideas in your product. The aims at this stage are to see if your ideas actually work and to show funders and/or company management the original product features that are better than those in competing products.
- ***Customer demonstration*** You take an existing prototype created to demonstrate feasibility and extend this with your ideas for specific customer features and how these can be realized. Before you develop this type of prototype, you need to do some user studies and have a clearer idea of your potential users and scenarios of use.

Anaylsis Part

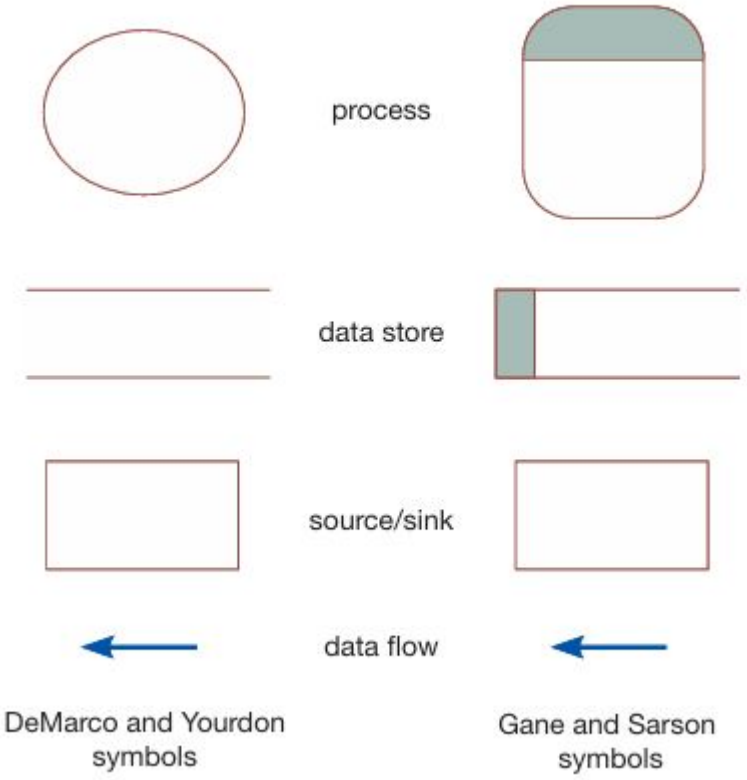
- Data Flow Diagramming Mechanics
- Business Process Modelling
- Activity Diagram

DATA FLOW DIAGRAMMING MECHANICS

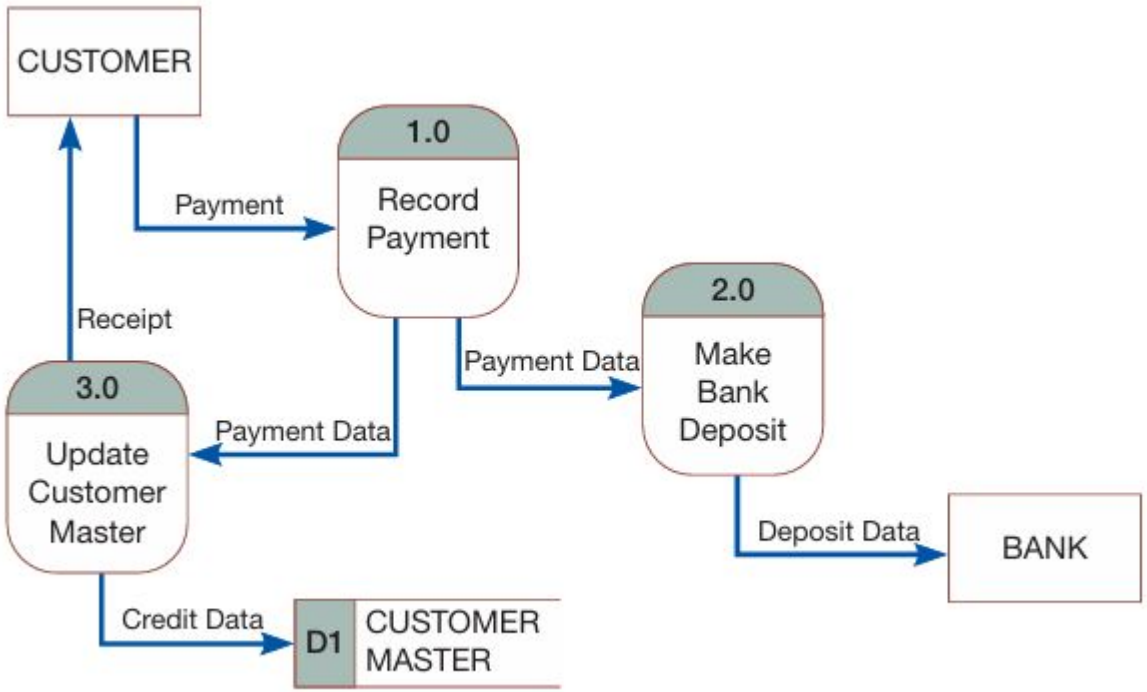
Data store
Data at rest, which may take the form of many different physical representations.

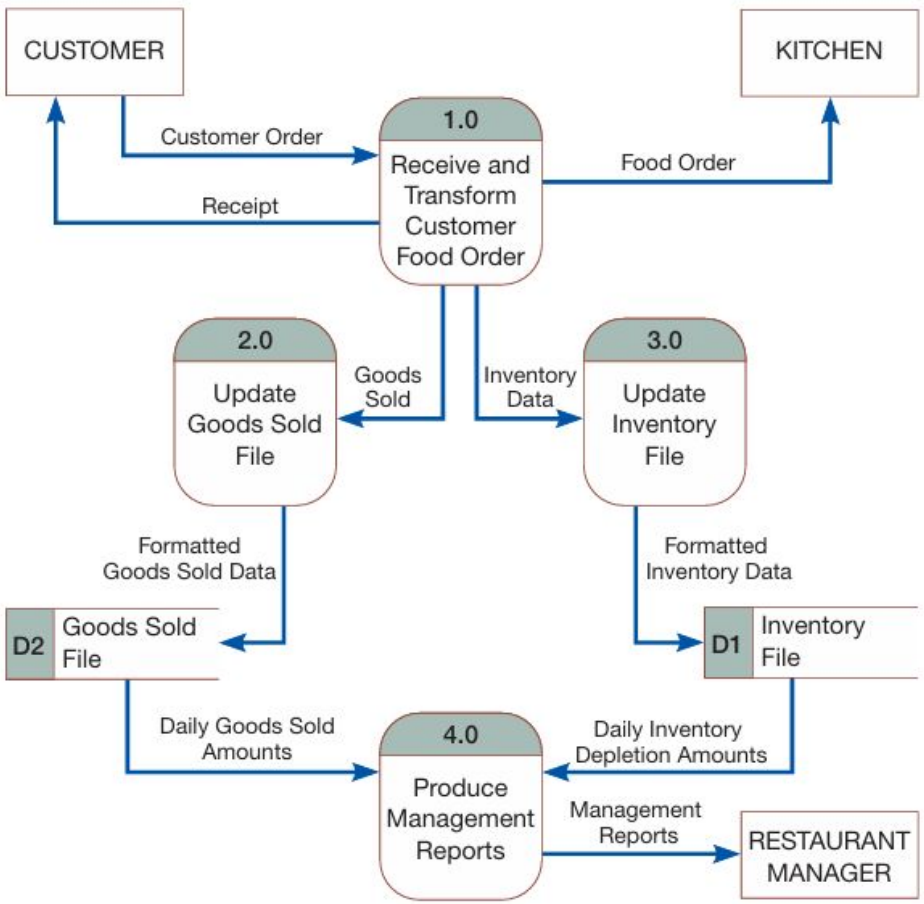
Process
The work or actions performed on data so that they are transformed, stored, or distributed.

Source/sink
The origin and/or destination of data; sometimes referred to as external entities.



Example


CUSTOMER
1.0
Record Payment
Payment
Receipt
2.0
Make Bank Deposit
Payment Data
3.0
Update Customer Master
Payment Data
Deposit Data
BANK
Credit Data
D1
CUSTOMER MASTER


CUSTOMER
KITCHEN
1.0
Receive and Transform Customer Food Order
Customer Order
Receipt
Food Order
2.0
Update Goods Sold File
3.0
Update Inventory File
Goods Sold
Inventory Data
Formatted Goods Sold Data
Formatted Inventory Data
D2
Goods Sold File
D1
Inventory File
Daily Goods Sold Amounts
Daily Inventory Depletion Amounts
4.0
Produce Management Reports
Management Reports
RESTAURANT MANAGER

Rules Governing Data Flow Diagramming

Process:

A. No process can have only outputs. It would be making data from nothing (a miracle). If an object has only outputs, then it must be a source.

B. No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.

C. A process has a verb phrase label.

Data Store:

D. Data cannot move directly from one data store to another data store. Data must be moved by a process.

E. Data cannot move directly from an outside source to a data store. Data must be moved by a process that receives data from the source and places the data into the data store.

F. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.

G. A data store has a noun phrase label.

Source/Sink:

H. Data cannot move directly from a source to a sink. It must be moved by a process if the data are of any concern to our system. Otherwise, the data flow is not shown on the DFD.

I. A source/sink has a noun phrase label.

Data Flow:

J. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows because these happen at different times.

K. A fork in a data flow means that exactly the same data goes from a common location to two or more different processes, data stores, or sources/sinks (this usually indicates different copies of the same data going to different locations).

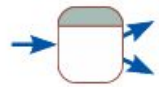
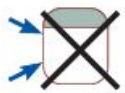
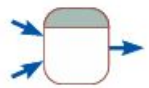
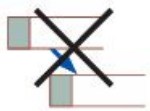
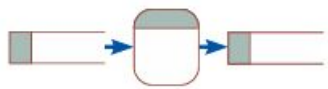
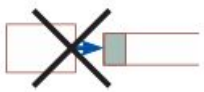
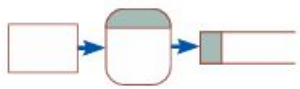
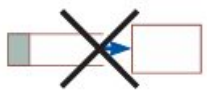
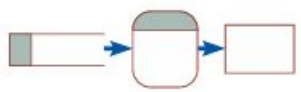
L. A join in a data flow means that exactly the same data come from any of two or more different processes, data stores, or sources/sinks to a common location.

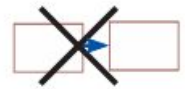
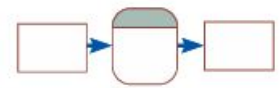
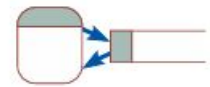
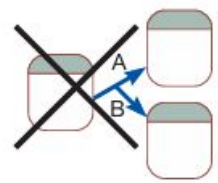
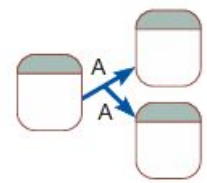
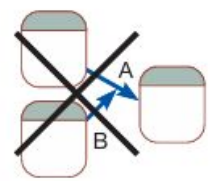
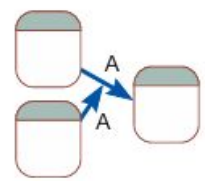
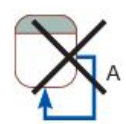
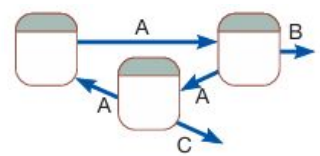
M. A data flow cannot go directly back to the same process it leaves. There must be at least one other process that handles the data flow, produces some other data flow, and returns the original data flow to the beginning process.

N. A data flow to a data store means update (delete or change).

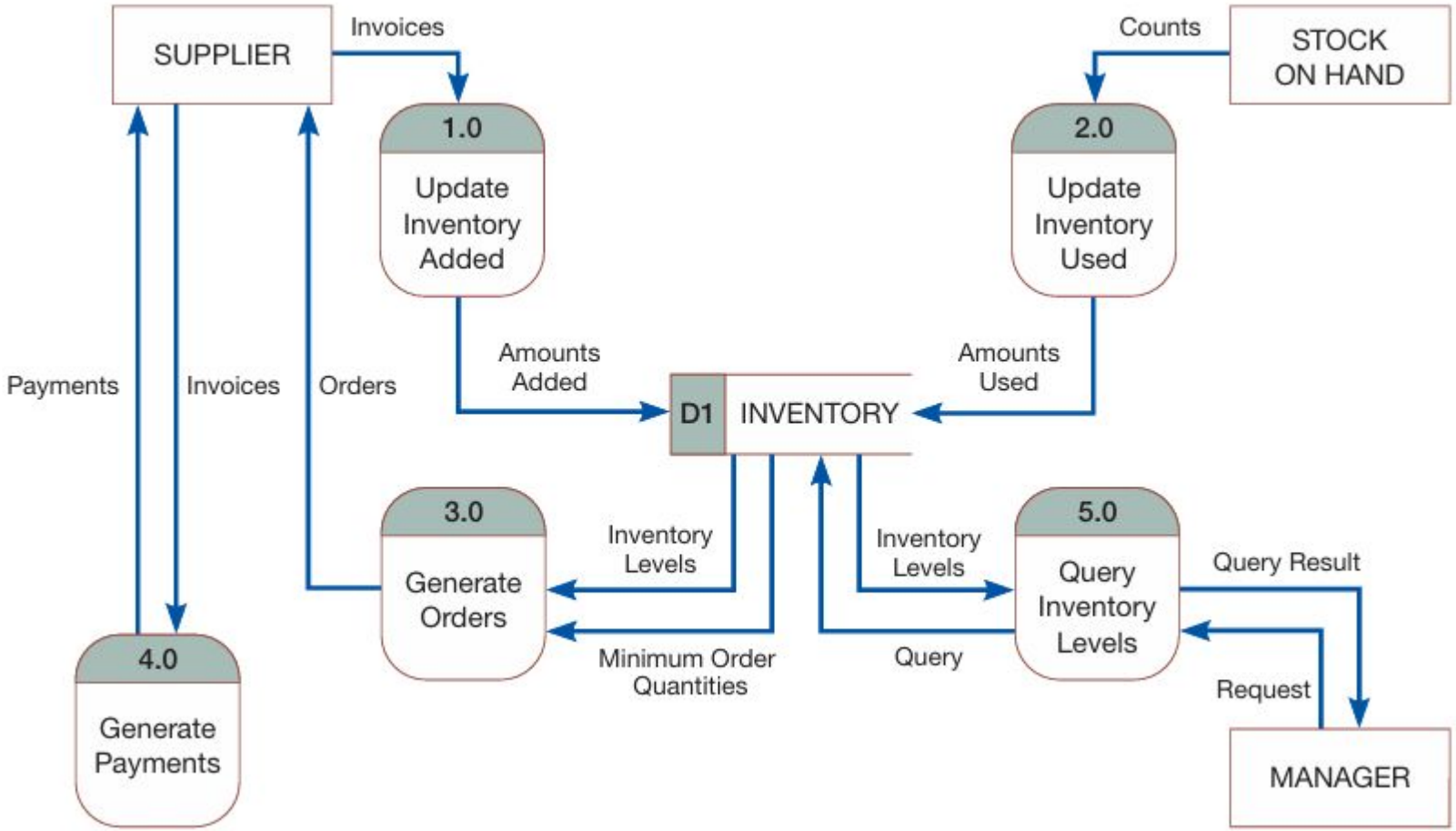
O. A data flow from a data store means retrieve or use.

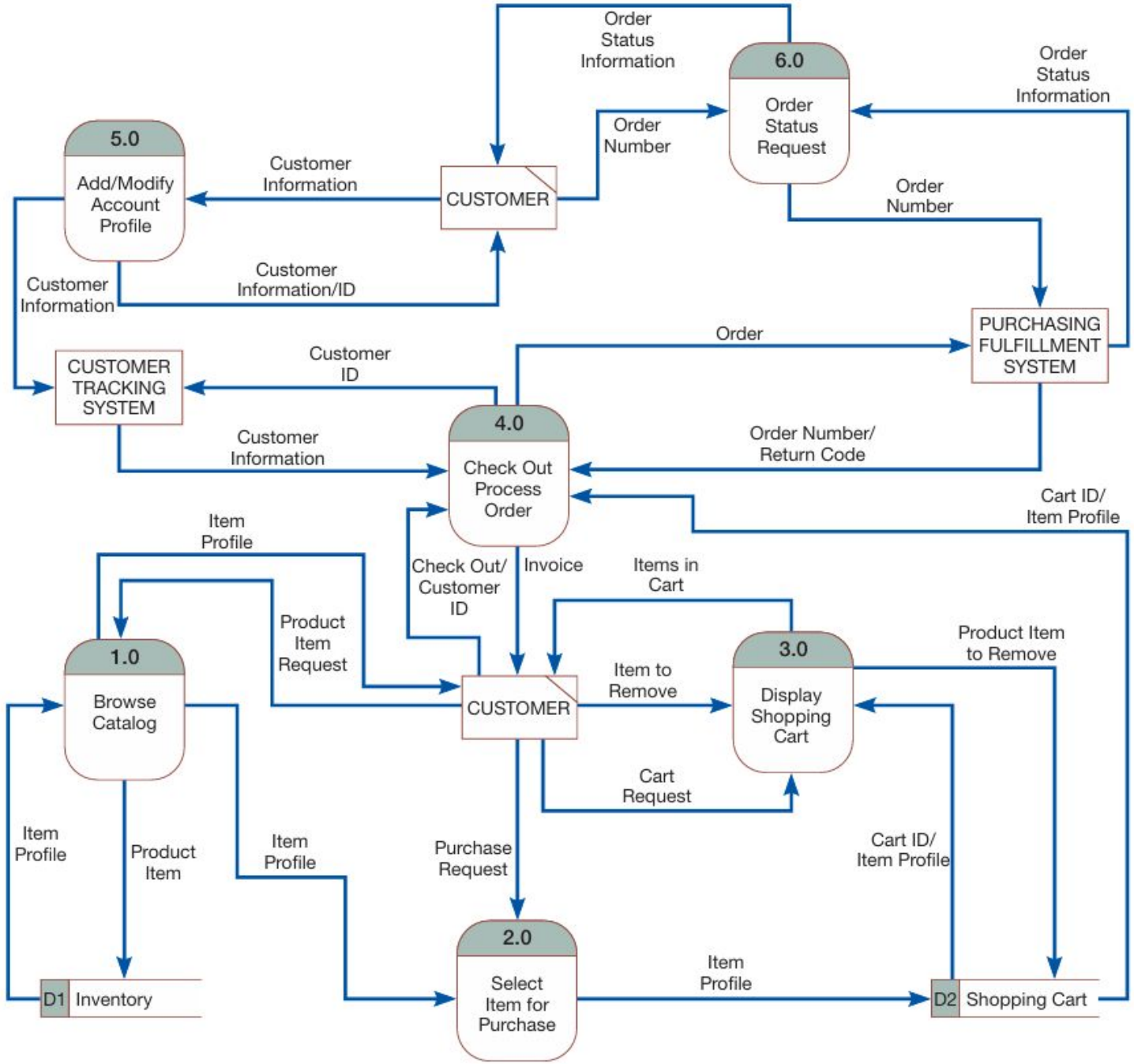
P. A data flow has a noun phrase label. More than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

Rule	Incorrect	Correct
A.		
B.		
D.		
E.		
F.		

Rule		
H.		
J.		
K.		
L.		
M.		

Advanced Example




Order Status Information
6.0
Order Status Request
Order Status Information
5.0
Add/Modify Account Profile
Customer Information
CUSTOMER
Order Number
Order Number
Customer Information
Customer Information/ID
PURCHASING FULFILLMENT SYSTEM
Order
CUSTOMER TRACKING SYSTEM
Customer ID
4.0
Check Out Process Order
Customer Information
Order Number/ Return Code
Cart ID/ Item Profile
Item Profile
Check Out/ Customer ID
Invoice
Items in Cart
Product Item Request
1.0
Browse Catalog
CUSTOMER
Item to Remove
3.0
Display Shopping Cart
Product Item to Remove
Cart Request
Item Profile
Product Item
Item Profile
Purchase Request
Cart ID/ Item Profile
2.0
Select Item for Purchase
Item Profile
D1 Inventory
D2 Shopping Cart

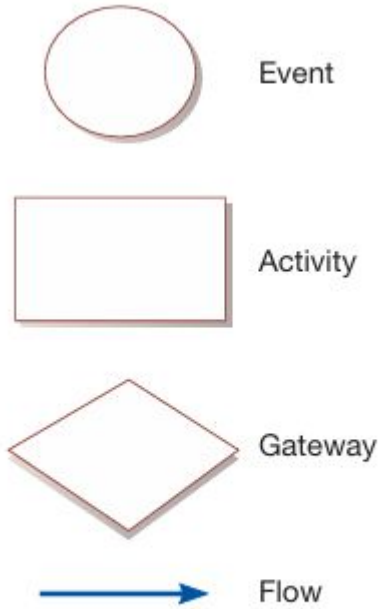
Busniess Process Modelling

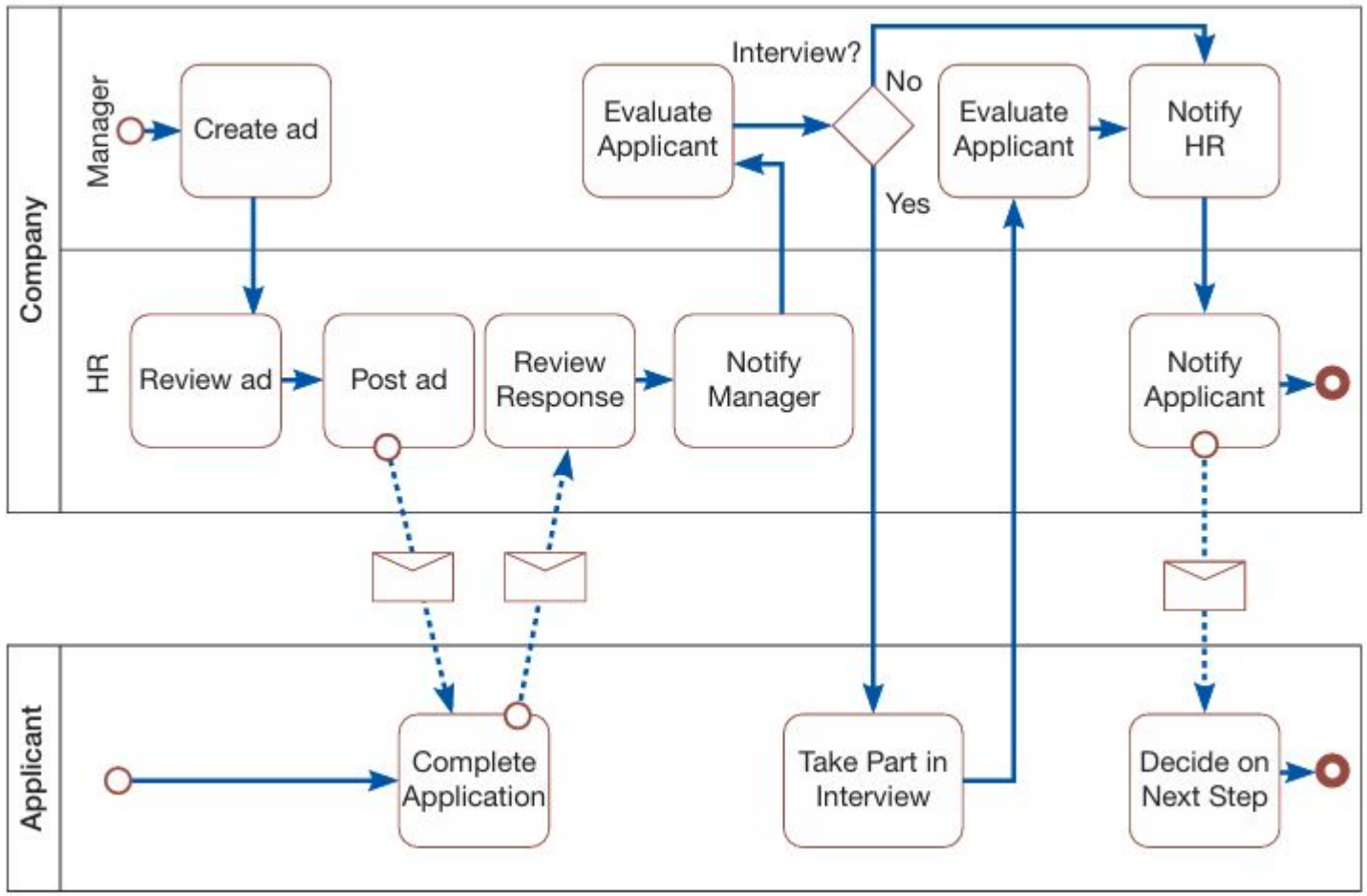
Event
In business process modeling, a trigger that initiates the start of a process.

Activity
In business process modeling, an action that must take place for a process to be completed.

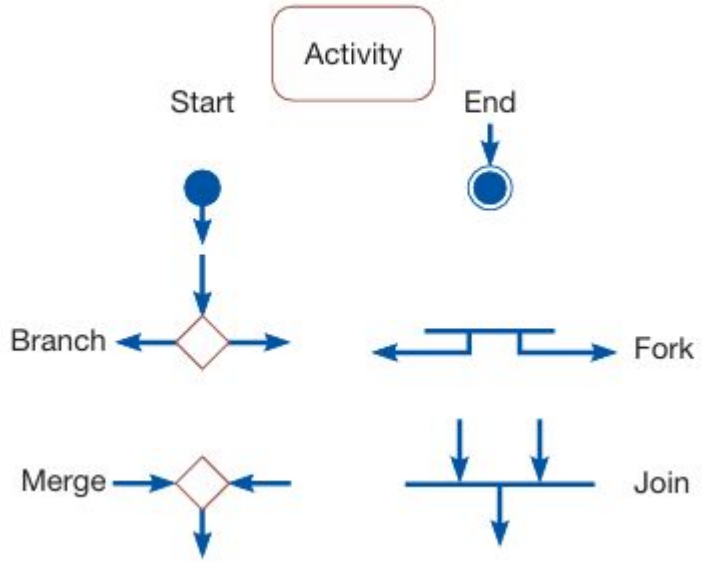
Gateway
In business process modeling, a decision point.

Flow
In business process modeling, it shows the sequence of action in a process.




Company
Manager
Create ad
Evaluate Applicant
Interview?
No
Yes
Evaluate Applicant
Notify HR
HR
Review ad
Post ad
Review Response
Notify Manager
Notify Applicant
Applicant
Complete Application
Take Part in Interview
Decide on Next Step

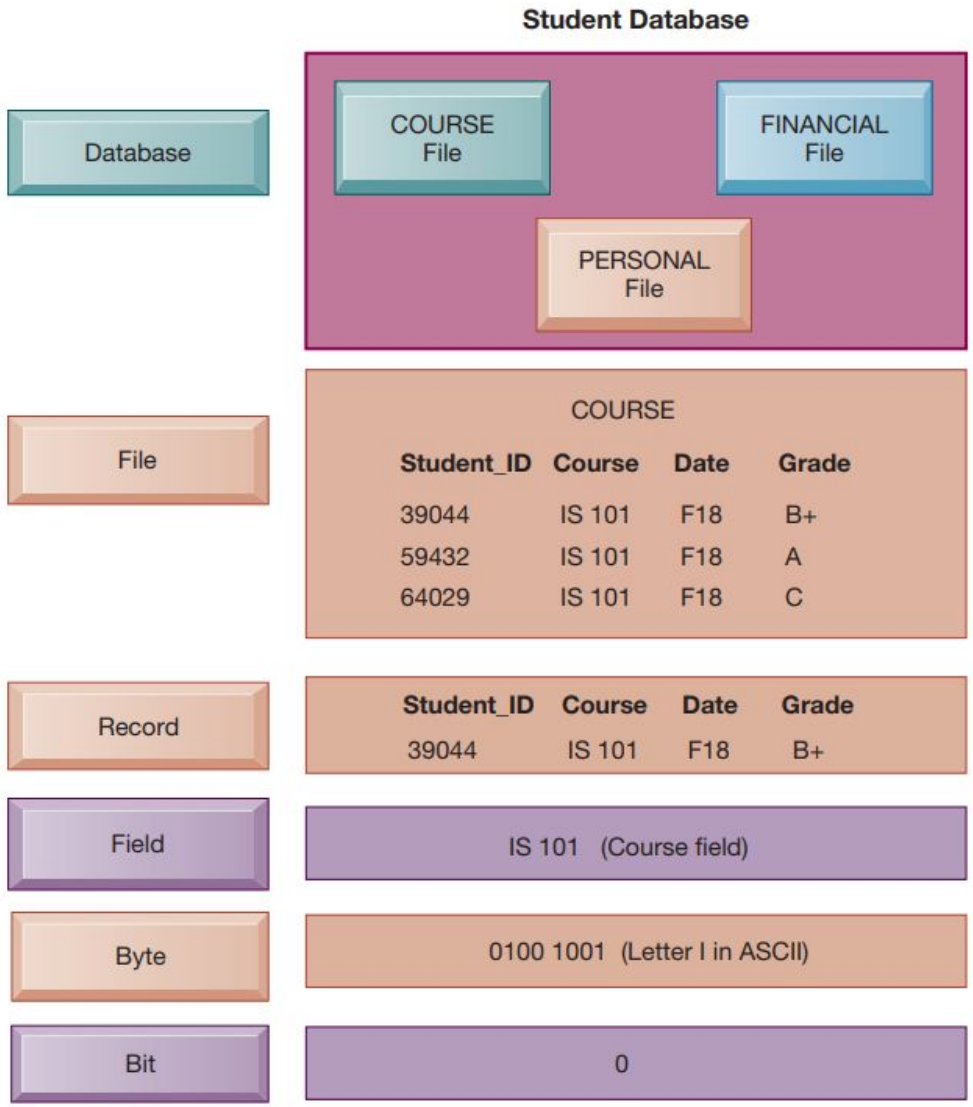
Activity Diagram



Mapping user requiremts to our tools

FIGURE 6.1 THE DATA HIERARCHY

A computer system organizes data in a hierarchy that starts with the bit, which represents either a 0 or a 1. Bits can be grouped to form a byte to represent one character, number, or symbol. Bytes can be grouped to form a field, and related fields can be grouped to form a record. Related records can be collected to form a file, and related files can be organized into a database.



You might also want to create a small, personal database using a software product such as Microsoft Access. In that case, you will need to be familiar with at least the basics of the product. **After the data are stored in your organization's databases, they must be accessible in a form that helps users to make decisions (Microsoft Power BI).**

Organizations accomplish this objective by developing data warehouses.

You should become familiar with data warehouses because they are invaluable decision-making tools.

We discuss data warehouses in Section 5.4.

Managing Data: the Difficulties of Managing Data

First, the amount of data increases exponentially with time. Much historical data must be kept for a long time, and new data are added rapidly. For example, to support millions of customers, large retailers such as Walmart have to manage many petabytes of data. (A petabyte is approximately 1,000 terabytes, or trillions of bytes.

Data are also scattered throughout organizations, and they are collected by many individuals using various methods and devices. These data are frequently stored in numerous servers and locations and in different computing systems, databases, formats, and human and computer languages.

- **Data rot refers primarily to problems with the media on which the data are stored.** Over time, temperature, humidity, and exposure to light can cause physical problems with storage media and thus make it difficult to access the data. The second aspect of data rot is that finding the machines needed to access the data can be difficult. For example, it is almost impossible today to find eight-track players to listen to music on. Consequently, a library of eight-track tapes has become relatively worthless, unless you have a functioning eight-track player or you convert the tapes to a more modern medium such as DVDs.

- Data security, quality, and integrity are critical, yet they are easily jeopardized. Legal requirements relating to data also differ among countries as well as among industries, and they change frequently.

Data Governance

- Data governance is an approach to managing information across an entire organization.
- Master data management is a process that spans all of an organization's business processes and applications. It provides companies with the ability to store, maintain, exchange, and synchronize a consistent, accurate, and timely "single version of the truth" for the company's master data
- Master data are a set of core data, such as customer, product, employee, vendor, geographic location, and so on, that span the enterprise's information systems. It is important to distinguish between master data and transactional data.
- Transactional data, which are generated and captured by operational systems, describe the business's activities, or transactions. In contrast, master data are applied to multiple transactions, and they are used to categorize, aggregate, and evaluate the transactional data.

Example: What are the main problem in the next table?

- Give an example of master data (student, course, student's marks) and you opened a file to store these data

Student Name	Student Number	Course ID	Course Name	Course Max	Course Pass	Term Year	Exam Date	Student degree
Ibrahim	6677	IS221	Information System	100	50	2020-2021 first term	15/01/2022	80
Mostafa	6678	IS221	Information System	100	50	2020-2021 first term	15/01/2022	90
Mennatallah	6678	IS221	Information System	100	50	2020-2021 first term	15/01/2022	95
Ibrahim	6677	IS221	Database	100	50	2020-2021 second term	15/06/2021	60
Mostafa	6677	IS222	Database	100	50	2020-2021 second term	15/06/2022	80

Here is the code

```
struct data{
char StudentName [50],
char StudentNumber[8],
...
float StudentDegree} ;

Data Students[1000]
```

```
void addStudent (i,Name,SN,...,degree)
{

stringcopy(student[i].StudentName,Name);
stringcopy(student[i].StudentNumber,SN);

...
student[i].StudentDegree = degree;


}
```

The main problem in file system and the advantage in database

- Data redundancy: The same data are stored in multiple locations.
- Data isolation: Applications cannot access data associated with other applications.
- Data inconsistency: Various copies of the data do not agree.

Database systems also maximize the following:

- Data security: Because data are “put in one place” in databases, there is a risk of losing a lot of data at one time. Therefore, databases must have extremely high-security measures in place to minimize mistakes and deter attacks.
- Data integrity: Data meet certain constraints; for example, there are no alphabetic characters in a Social Insurance Number field.
- Data independence: Applications and data are independent of one another; that is, applications and data are not linked to each other, so all applications are able to access the same data

Database management system

A database management system (DBMS) is a set of programs that provide users with tools to create and manage a database. Managing a database refers to the processes of adding, deleting, accessing, modifying, and analyzing data that are stored in a database. An organization can access these data by using query and reporting tools that are part of the DBMS or by utilizing application programs specifically written to perform this function. DBMSs also provide the
mechanisms for maintaining the integrity of stored data, managing security and user access, and recovering information if the system fails. Because databases and DBMSs are essential to all areas of business, they must be carefully managed.

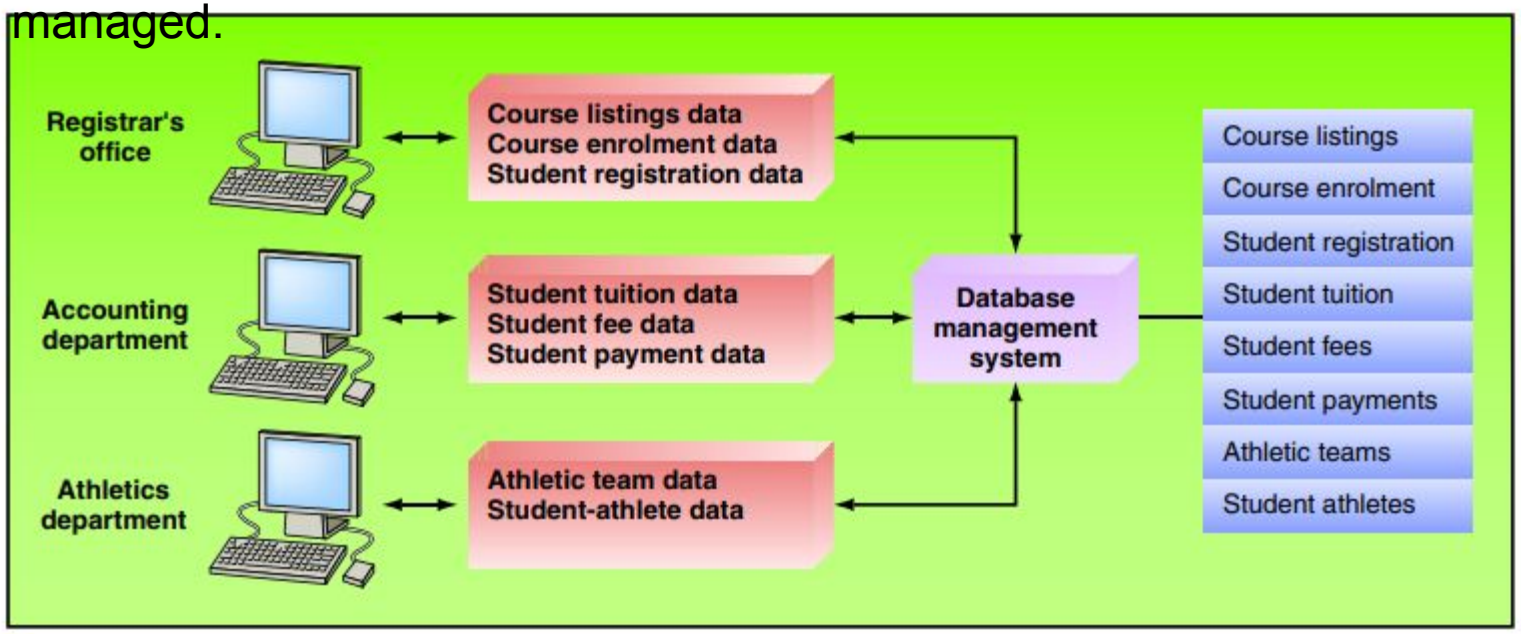


FIGURE 5.1 Database management system.

- An entity is a database table
- data model is a diagram that represents entities in the database and their relationships
- An instance is a row in a database table.
- An Attribute is a field name, such as studentName, studentDegree
- primary key is an attribute having two constraints:
  a unique value
  index

- A foreign key is …

Query Languages

- The most commonly performed database operation is searching for information. Structured query language (SQL) is the most popular query language

```
SELECT Student_Name
FROM Student_Database
WHERE Grade_Point_Average > = 3.40 and Grade_Point_Average < 3.60;
```

Data Warehouses and Data Marts

Now, the company built the used it for 10 years.

The challenge is to provide users with access to corporate data so they can analyze the data to make better decisions.

if a manager needed to know the trend in the profit margins on used books over the past 10 years, then he/she would have to construct a very complicated SQL or QBE query.

The complicated queries might take a long time to answer, and they also might degrade the performance of the databases.

The reasons are that:
transactional databases are designed to access a single record at a time. In contrast, we need to access large groups of related records.

Data warehouse and data marts

- A data warehouse is a repository of historical data that are organized by subject to support decision makers within the organization.
- A data mart is a low-cost, scaled-down version of a data warehouse that is designed for the end-user needs in a strategic business unit (SBU) or an individual department.

- Typically, groups that need a single or a few business analytics applications require only a data mart rather than a data warehouse.

What do Dataware reprsent?

(a) 2018

Product	Region	Sales
Nuts	East	50
Nuts	West	60
Nuts	Central	100
Screws	East	40
Screws	West	70
Screws	Central	80
Bolts	East	90
Bolts	West	120
Bolts	Central	140
Washers	East	20
Washers	West	10
Washers	Central	30

(b) 2019

Product	Region	Sales
Nuts	East	60
Nuts	West	70
Nuts	Central	110
Screws	East	50
Screws	West	80
Screws	Central	90
Bolts	East	100
Bolts	West	130
Bolts	Central	150
Washers	East	30
Washers	West	20
Washers	Central	40

(c) 2020

Product	Region	Sales
Nuts	East	70
Nuts	West	80
Nuts	Central	120
Screws	East	60
Screws	West	90
Screws	Central	100
Bolts	East	110
Bolts	West	140
Bolts	Central	160
Washers	East	40
Washers	West	30
Washers	Central	50

FIGURE 5.5 **Relational databases.**

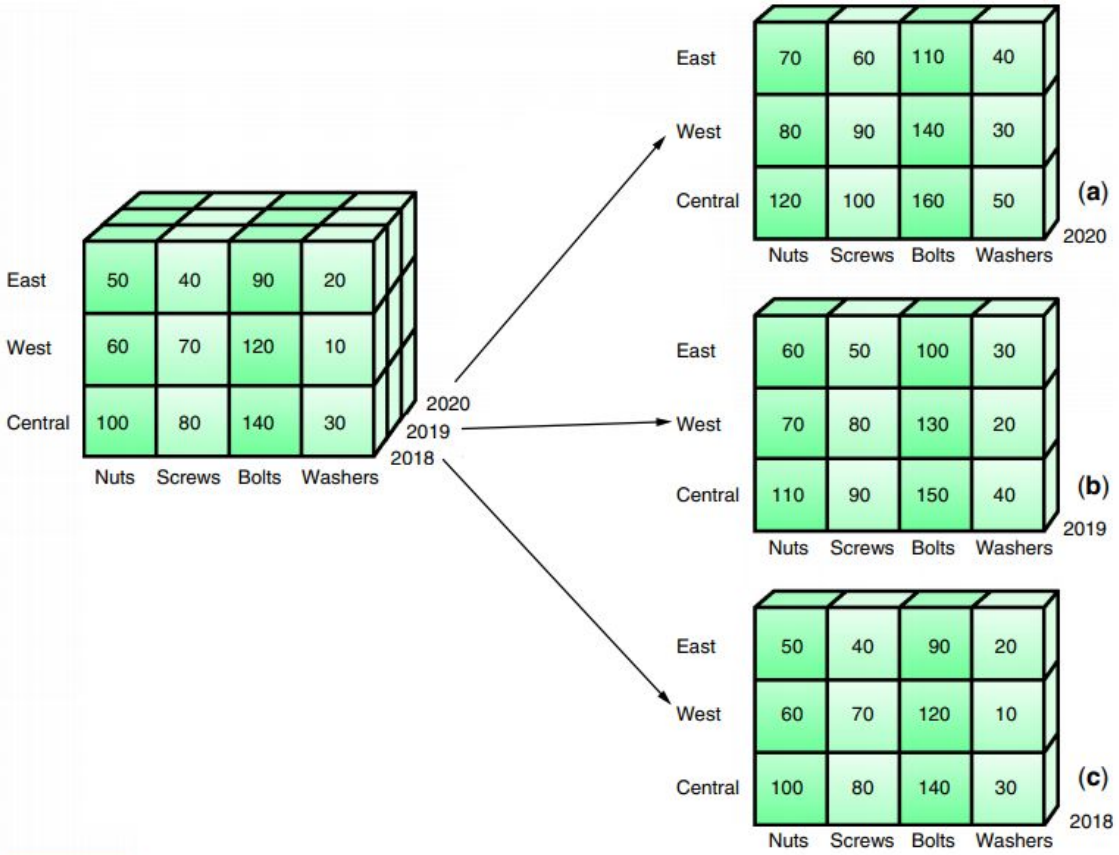


FIGURE 5.6 **Data cube.**

Data warehouse framework.

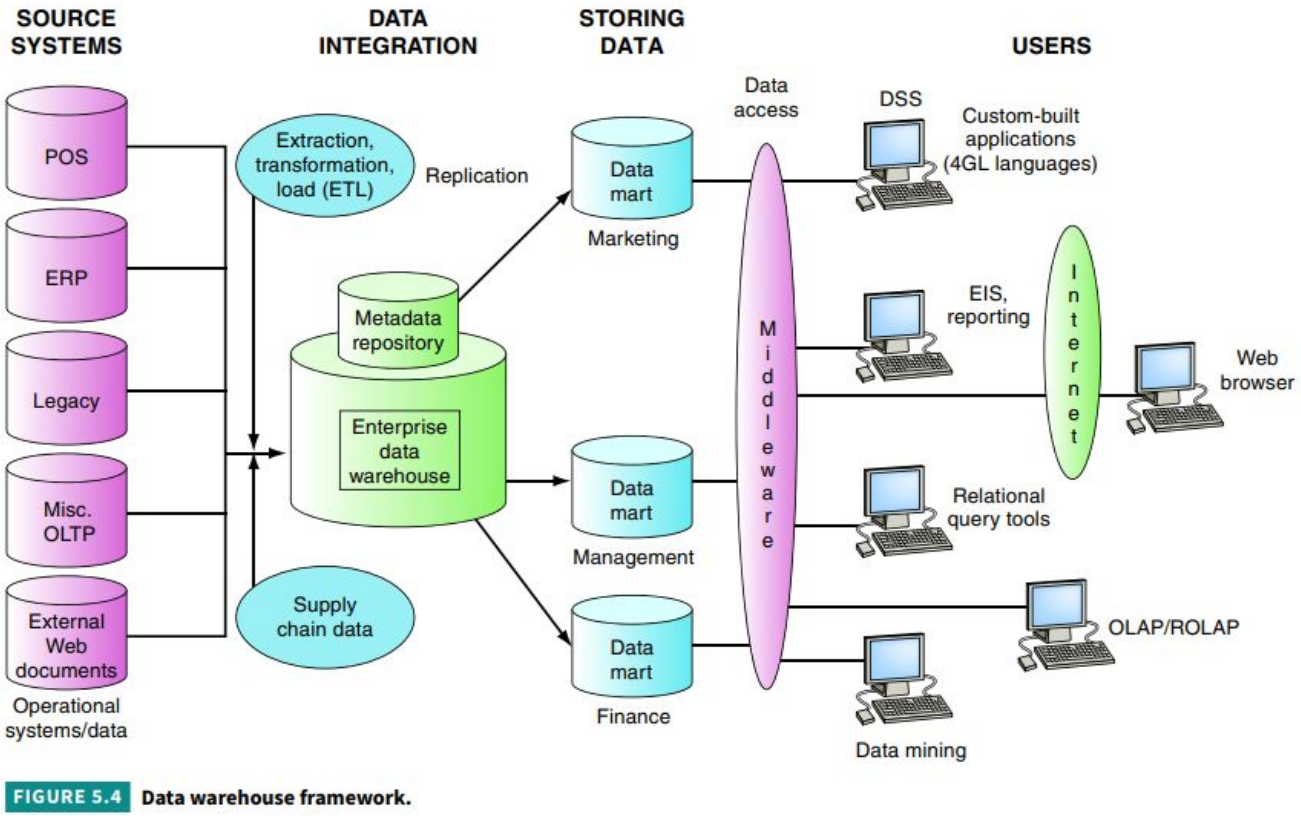


FIGURE 5.4 Data warehouse framework.

The benefits of data warehousing include the following:

- End users can access needed data quickly and easily through web browsers because these data are located in one place.
- End users can conduct extensive analysis with data in ways that were not previously possible.
- End users can obtain a consolidated view of organizational data.

Metadata is important to maintain data about the data, known as metadata, in the data warehouse. Both the IT personnel who operate and manage the data warehouse and the users who access the data require metadata.

Users Once the data are loaded in a data mart or warehouse, they can be accessed. At this point, the organization begins to obtain business value from BI; all of the prior stages constitute creating BI infrastructure.

Storing the Data Organizations can choose from a variety of architectures to store decision-support data. The most common architecture is one central enterprise data warehouse, without data marts. Most organizations use this approach because the data stored in the warehouse are accessed by all users, and they represent the single version of the truth.

Data warehouse framework.

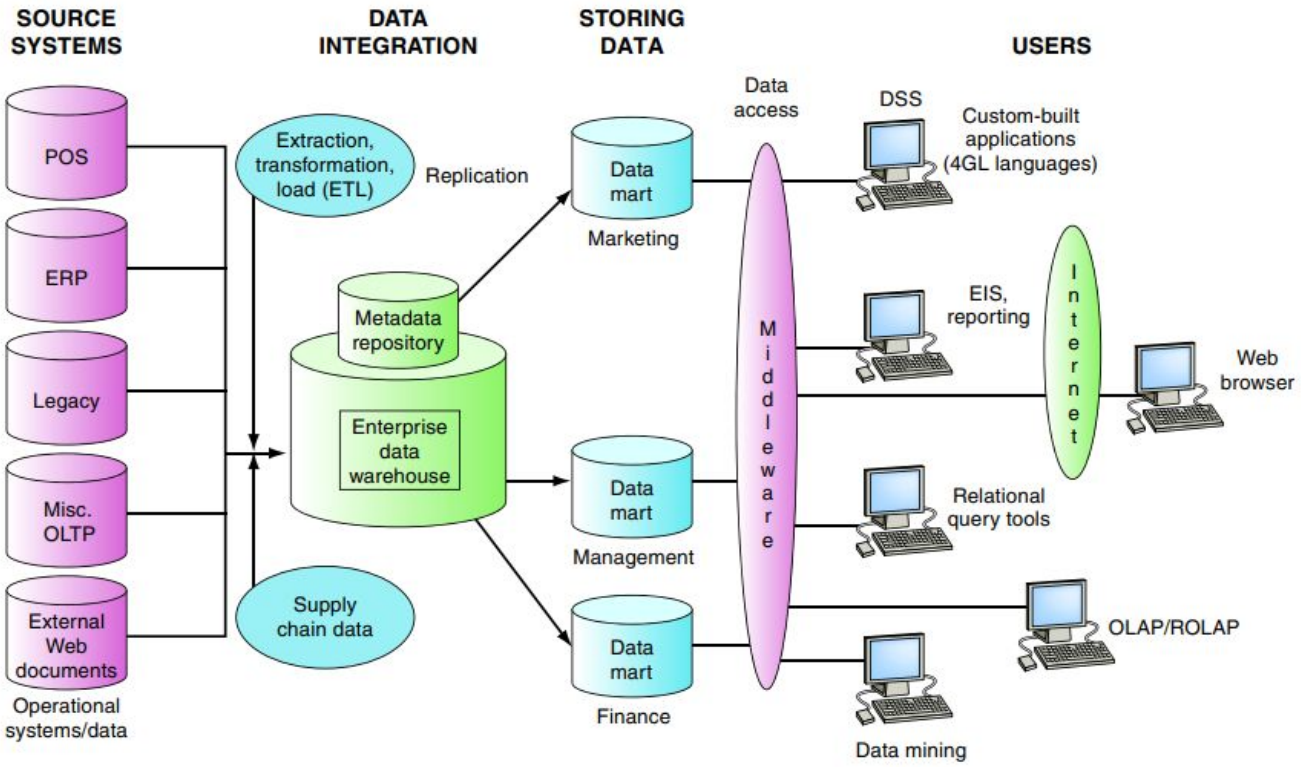


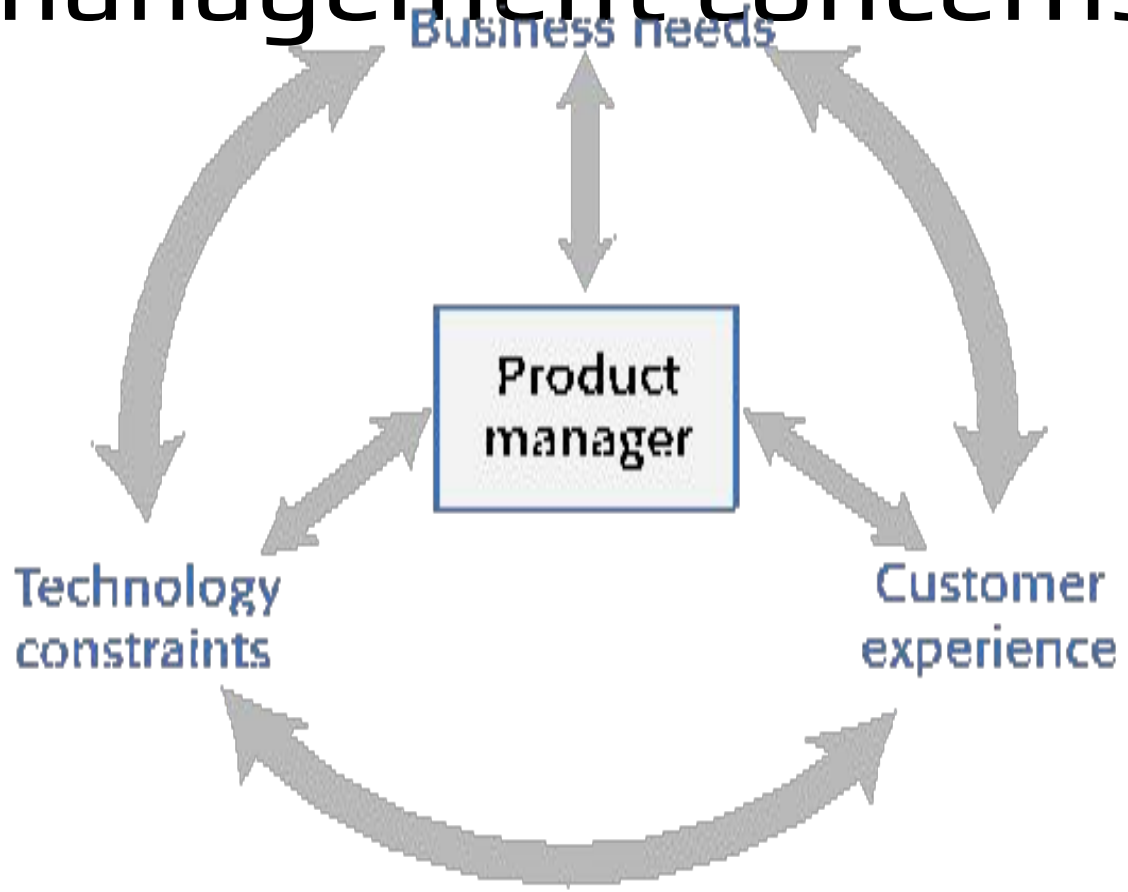
FIGURE 5.4 Data warehouse framework.

Data Quality The quality of the data in the warehouse must meet users' needs. If it does not, then users will not trust the data and ultimately will not use it. Most organizations find that the quality of the data in source systems is poor and must be improved before the data can be used in the data warehouse. Some of the data can be improved with data-cleansing software.

Governance To ensure that BI is meeting their needs, organizations must implement governance to plan and control their BI activities. Governance requires that people, committees, and processes be in place. Companies that are effective in BI governance often create a senior-level committee composed of vice presidents and directors who (1) ensure that the business strategies and BI strategies are in alignment, (2) prioritize projects, and (3) allocate resources.

These companies also establish a middle-management–level committee that oversees the various projects in the BI portfolio to ensure that these projects are being completed in accordance with the company's objectives. Finally, lower-level operational committees perform tasks such as creating data definitions and identifying and solving data problems. All of these committees rely on the collaboration and contributions of business users and IT personnel.

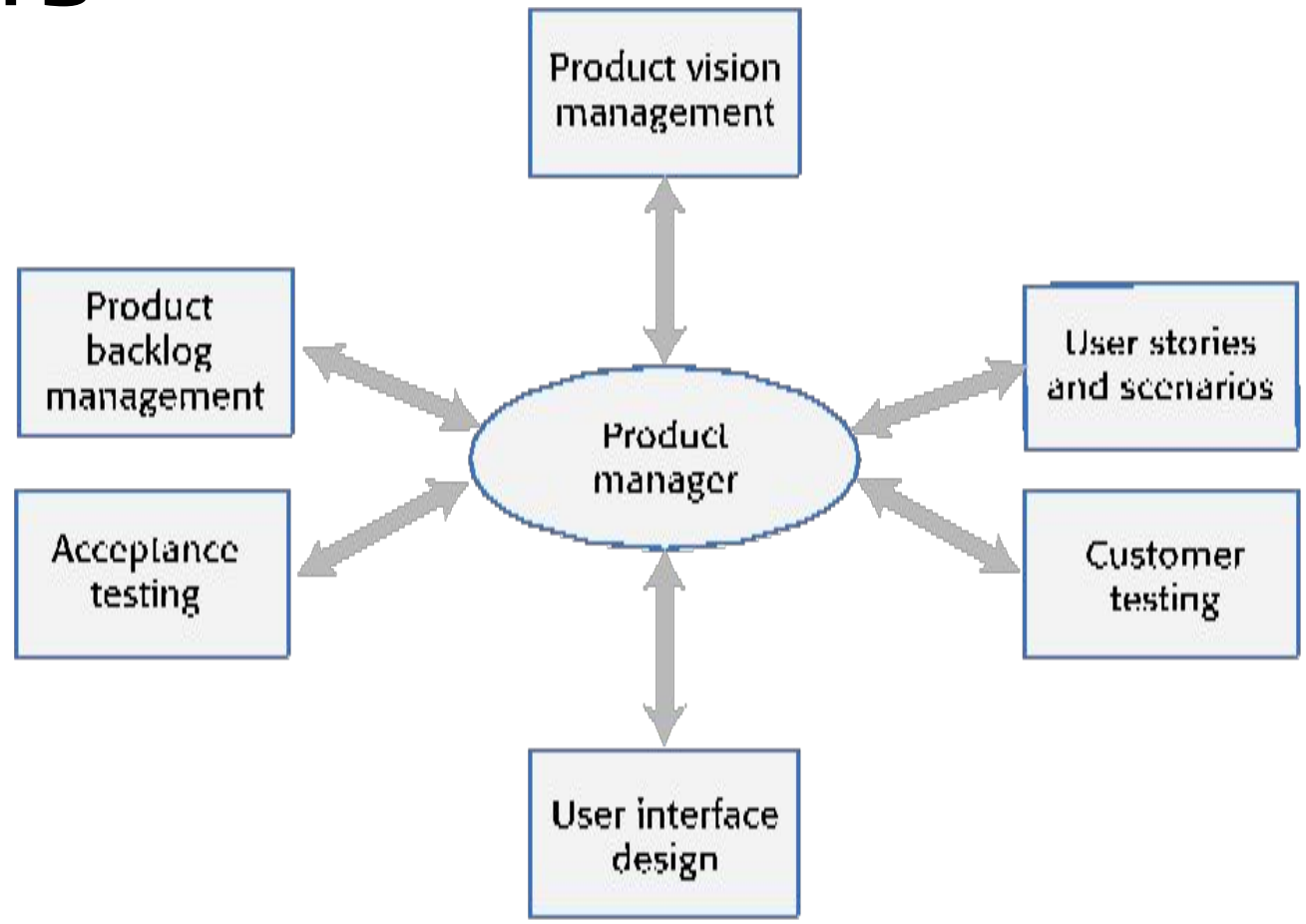
Product management concerns



Product management concerns

- ***Business needs*** PMs have to ensure that the software being developed meets the business goals of the software development company.
- ***Technology constraints*** PMs must make developers aware of technology issues that are important to customers.
- ***Customer experience*** PMs should be in regular contact with customers and potential customers to understand what they are looking for in a product, the types of users and their backgrounds and the ways that the product may be used.

Technical interactions of product managers



Technical interactions of product managers

- Product vision management
  - The product manager may be responsible for helping with the development of the product vision. The should always be responsible for managing the vision, which involves assessing and evaluating proposed changes against the product vision. They should ensure that there is no ‘vision drift’
- Product roadmap development
  - A product roadmap is a plan for the development, release and marketing of the software. The PM should lead roadmap development and should be the ultimate authority in deciding if changes to the roadmap should be made.
- User story and scenario development
  - User stories and scenarios are used to refine a product vision and identify product features. Based on his or her knowledge of customers, the PM should lead the development of stories and scenarios.

Technical interactions of product managers

- Product backlog creation and management
  - The product backlog is a prioritized ‘to-do’ list of what has to be developed. PMs should be involved in creating and refining the backlog and deciding on the priority of product features to be developed.
- Acceptance testing
  - Acceptance testing is the process of verifying that a software release meets the goals set out in the product roadmap and that the product is efficient and reliable. The PM should be involved in developing tests of the product features that reflect how customers use the product.
- Customer testing
  - Customer testing involves taking a release of a product to customers and getting feedback on the product’s features, usability and business. PMs are involved in selecting customers to be involved in the customer testing process and working with them during that process.
- User interface design
  - Product managers should understand user limitations and act as surrogate users in their interactions with the development team. They should evaluate user interface features as they are developed to check that these features are not unnecessarily complex or force users to work in an unnatural way.

Product prototyping

- Product prototyping is the process of developing an early version of a product to test your ideas and to convince yourself and company funders that your product has real market potential.
  - You may be able to write an inspiring product vision, but your potential users can only really relate to your product when they see a working version of your software. They can point out what they like and don't like about it and make suggestions for new features.
  - A prototype may be also used to help identify fundamental software components or services and to test technology.
- Building a prototype should be the first thing that you do when developing a software product. Your aim should be to have a working version of your software that can be used to demonstrate its key features.
- You should always plan to throw-away the prototype after development and to re-implement the software, taking account of issues such as security and reliability.

Two-stage prototyping

- ***Feasibility demonstration*** You create an executable system that demonstrates the new ideas in your product. The aims at this stage are to see if your ideas actually work and to show funders and/or company management the original product features that are better than those in competing products.
- ***Customer demonstration*** You take an existing prototype created to demonstrate feasibility and extend this with your ideas for specific customer features and how these can be realized. Before you develop this type of prototype, you need to do some user studies and have a clearer idea of your potential users and scenarios of use.

Design pattern definition

- Definition
  - **A general reusable solution to a commonly-occurring problem within a given context in software design.**
- Design patterns are object-oriented and describe solutions in terms of objects and classes. They are not off-the-shelf solutions that can be directly expressed as code in an object-oriented language.
- They describe the structure of a problem solution but have to be adapted to suit your application and the programming language that you are using.

Programming principles

- Separation of concerns
  - This means that each abstraction in the program (class, method, etc.) should address a separate concern and that all aspects of that concern should be covered there. For example, if authentication is a concern in your program, then everything to do with authentication should be in one place, rather than distributed throughout your code.
- Separate the ‘what’ from the ‘how
  - If a program component provides a particular service, you should make available only the information that is required to use that service (the ‘what’). The implementation of the service (‘the how’) should be of no interest to service users.

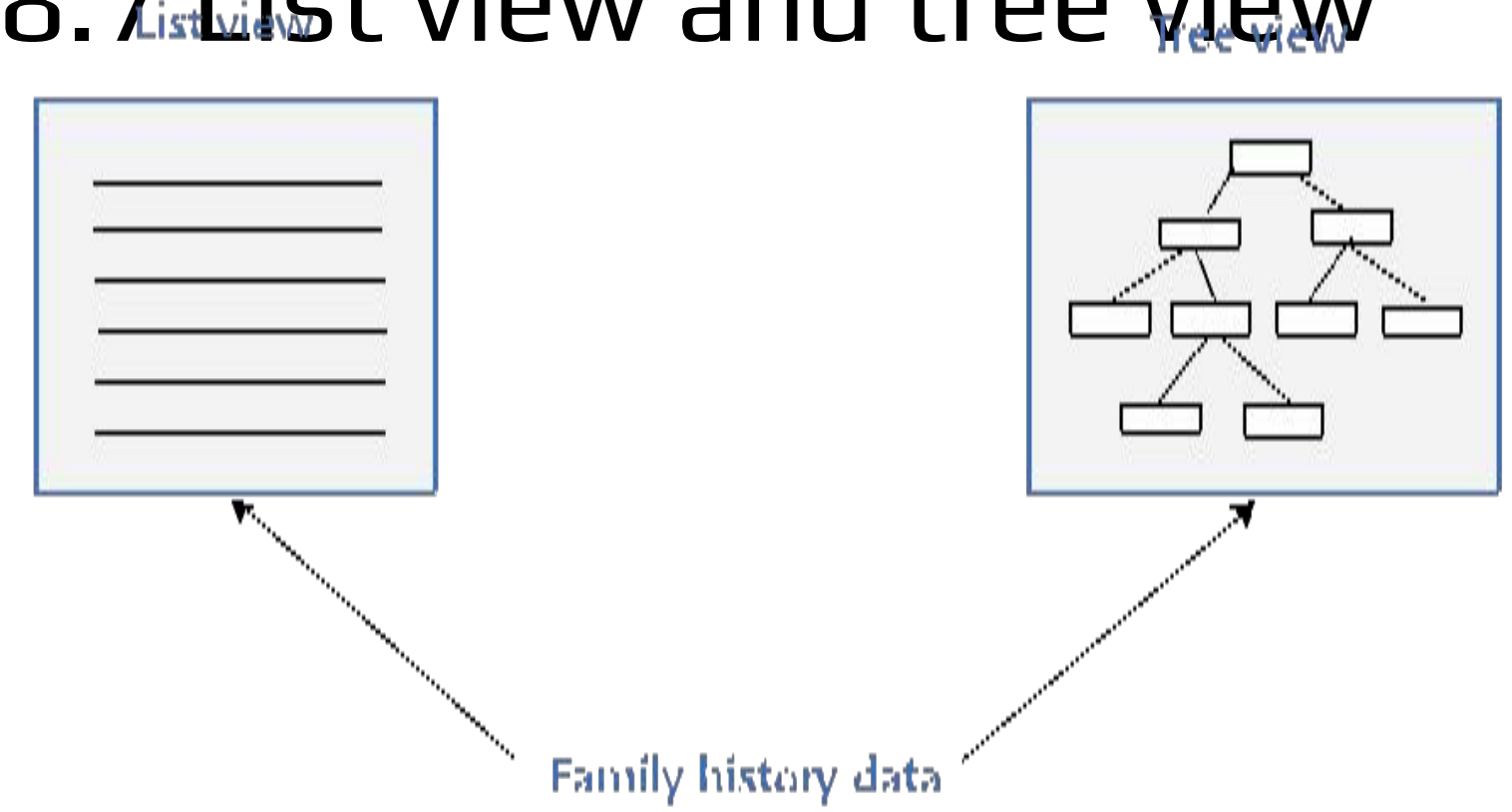
Common types of design patterns

- Creational patterns
  - These are concerned with class and object creation. They define ways of instantiating and initializing objects and classes that are more abstract than the basic class and object creation mechanisms defined in a programming language.
- Structural patterns
  - These are concerned with class and object composition. Structural design patterns are a description of how classes and objects may be combined to create larger structures.
- Behavioural patterns
  - These are concerned with class and object communication. They show how objects interact by exchanging messages, the activities in a process and how these are distributed amongst the participating objects.

Table 8.2 Examples of design patterns

Pattern name	*Type*	*Description*
Factory	Creational	Used to create objects when slightly different variants of the object may be created.
Prototype	Creational	Used to create an object clone i.e. a new object with exactly the same attribute values as the object being cloned.
Adapter	Structural	Used to match semantically-compatible interfaces of different classes.
Facade	Structural	Used to provide a single interface to a group of classes in which each class implements some functionality accessed through the interface.
Mediator	Behavioural	Used to reduce the number of direct interactions between objects. All object communications are handled through the mediator.
State	Behavioural	Used to implement a state machine where the behaviour of an object when its internal state changes.

Figure 8.7 List view and tree view



Presenting multiple views of the same data

Table 8.3 The Observer pattern (1)

- *Description*
  This pattern separates the display of an object from the object itself. There may be multiple displays associated with the object. When one display is changed, all others are notified and take action to update themselves.
- *Problem*
  Many applications present multiple views (displays) of the same data with the requirement that all views must be updated when any one view is changed. You may also wish to add new views without the object, whose state is being displayed, knowing about the new view or how the information is presented.
- *Solution*
  The state to be displayed (sometimes called the Model) is maintained in a Subject class that includes methods to add and remove observers and to get and set the state of the Model. An observer is created for each display and registers with the Subject. When an observer uses the set method to change the state, the Subject notifies all other Observers. They then use the Subject's getState( ) method to update their local copy of the state and so change their display. Adding a new display simply involves notifying the Subject that a new display has been created.

Figure 8.3 The Observer pattern (2)

- *Implementation*
  This pattern is implemented using abstract and concrete classes. The abstract Subject class includes methods to register and deregister observers and to notify all observers that a change has been made. The abstract Observer class includes a method to update the local state of each observer. Each Observer subclass implements these methods and is responsible for managing its own display. When notifications of a change are received, the Observer subclasses access the model using the getState( ) method to retrieve the changed information.
- *Things to consider*
  The Subject does not know how the Model is displayed so cannot organize its data to optimize the display performance. If a display update fails, the Subject does not know that the update has been unsuccessful.

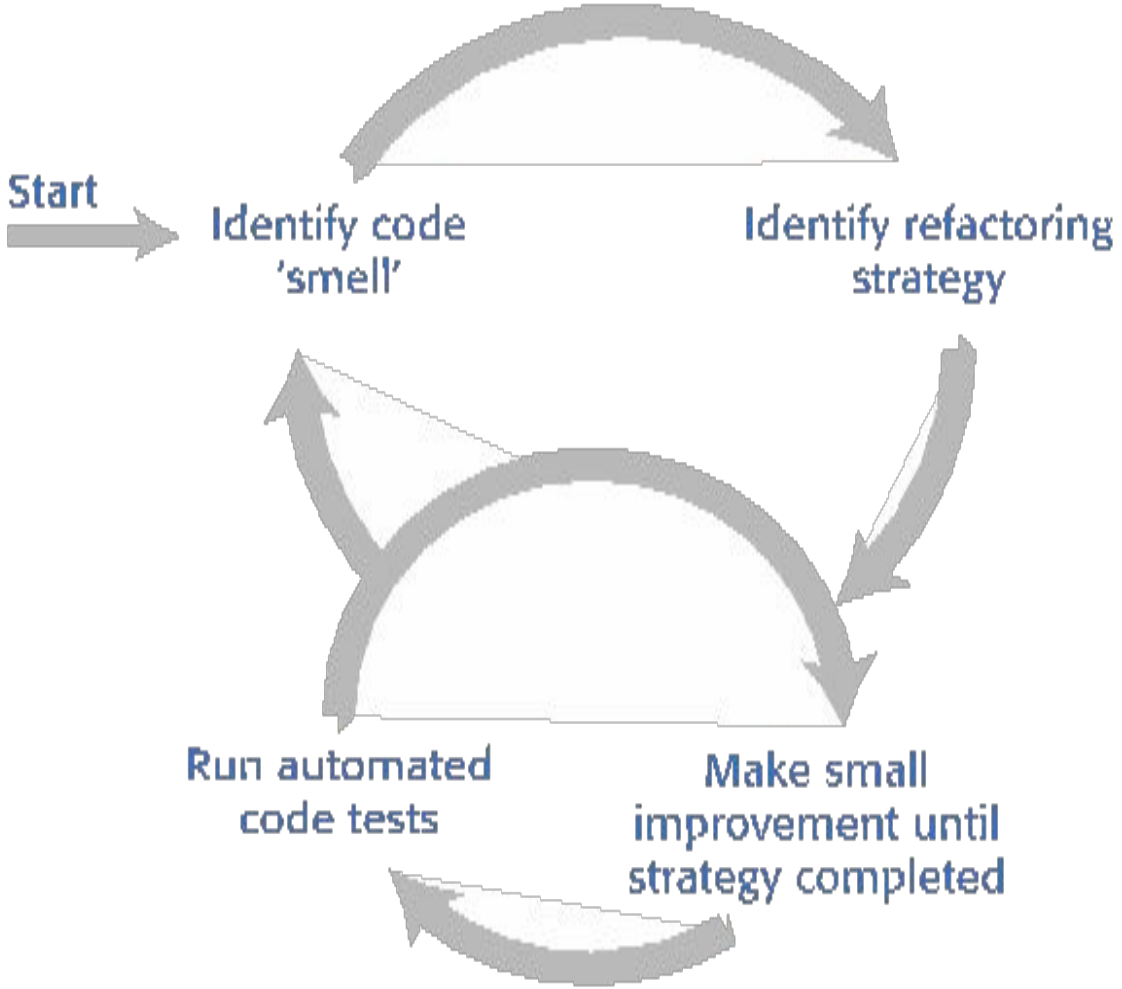
Pattern description

- Design patterns are usually documented in the stylized way. This includes:
  - a meaningful name for the pattern and a brief description of what it does;
  - a description of the problem it solves;
  - a description of the solution and its implementation;
  - the consequences and trade-offs of using the pattern and other issues that you should consider.

Refactoring

- Refactoring means changing a program to reduce its complexity without changing the external behaviour of that program.
- Refactoring makes a program more readable (so reducing the ‘reading complexity’) and more understandable.
- It also makes it easier to change, which means that you reduce the chances of making mistakes when you introduce new features.
- The reality of programming is that as you make changes and additions to existing code, you inevitably increase its complexity.
  - The code becomes harder to understand and change. The abstractions and operations that you started with become more and more complex because you modify them in ways that you did not originally anticipate.

Figure 8.8 A refactoring process



Code smells

- Martin Fowler, a refactoring pioneer, suggests that the starting point for refactoring should be to identify code ‘smells’.
- Code smells are indicators in the code that there might be a deeper problem.
  - For example, very large classes may indicate that the class is trying to do too much. This probably means that its structural complexity is high.

Table 8.6 Examples of code smells

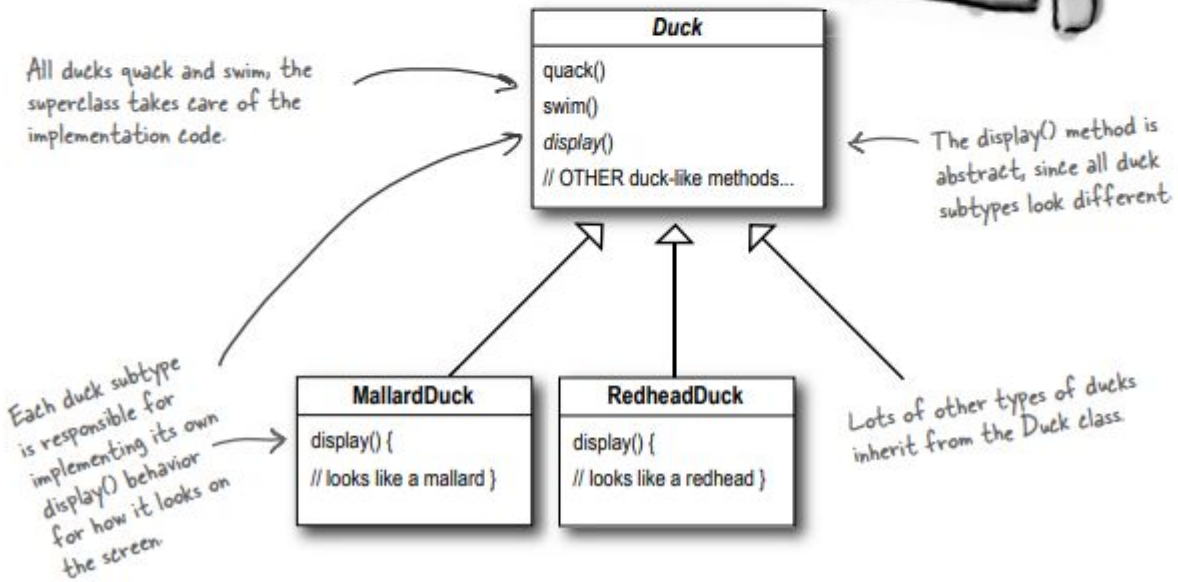
- *Large classes*
  Large classes may mean that the single responsibility principle is being violated. Break down large classes into easier-to-understand, smaller classes.
- *Long methods/functions*
  Long methods or functions may indicate that the function is doing more than one thing. Split into smaller, more specific functions or methods.
- *Duplicated code*
  Duplicated code may mean that when changes are needed, these have to be made everywhere the code is duplicated. Rewrite to create a single instance of the duplicated code that is used as required
- *Meaningless names*
  Meaningless names are a sign of programmer haste. They make the code harder to understand. Replace with meaningful names and check for other shortcuts that the programmer may have taken.
- *Unused code*
  This simply increases the reading complexity of the code. Delete it even if it has been commented out. If you find you need it later, you should be able to retrieve it from the code management system.

Examples of refactoring for complexity reduction

- *Reading complexity*
  You can rename variable, function and class names throughout your program to make their purpose more obvious.
- *Structural complexity*
  You can break long classes or functions into shorter units that are likely to be more cohesive than the original large class.
- *Data complexity*
  You can simplify data by changing your database schema or reducing its complexity. For example, you can merge related tables in your database to remove duplicated data held in these tables.
- *Decision complexity*
  You can replace a series of deeply nested if-then-else statements with guard clauses, as I explained earlier in this chapter.

It started with a simple SimUDuck app

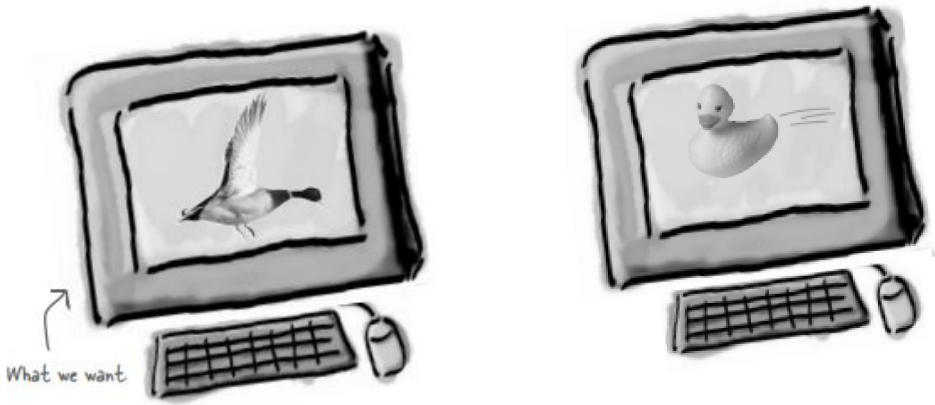
Joe works for a company that makes a highly successful duck pond simulation game, *SimUDuck*. The game can show a large variety of duck species swimming and making quacking sounds. The initial designers of the system used standard OO techniques and created one Duck superclass from which all other duck types inherit.



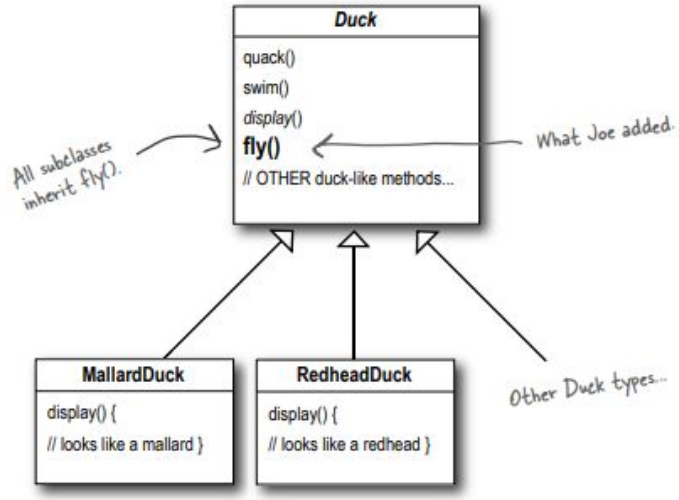
Mallard Duck

Redhead Duck

What is happen, if we include new function of flying of durk

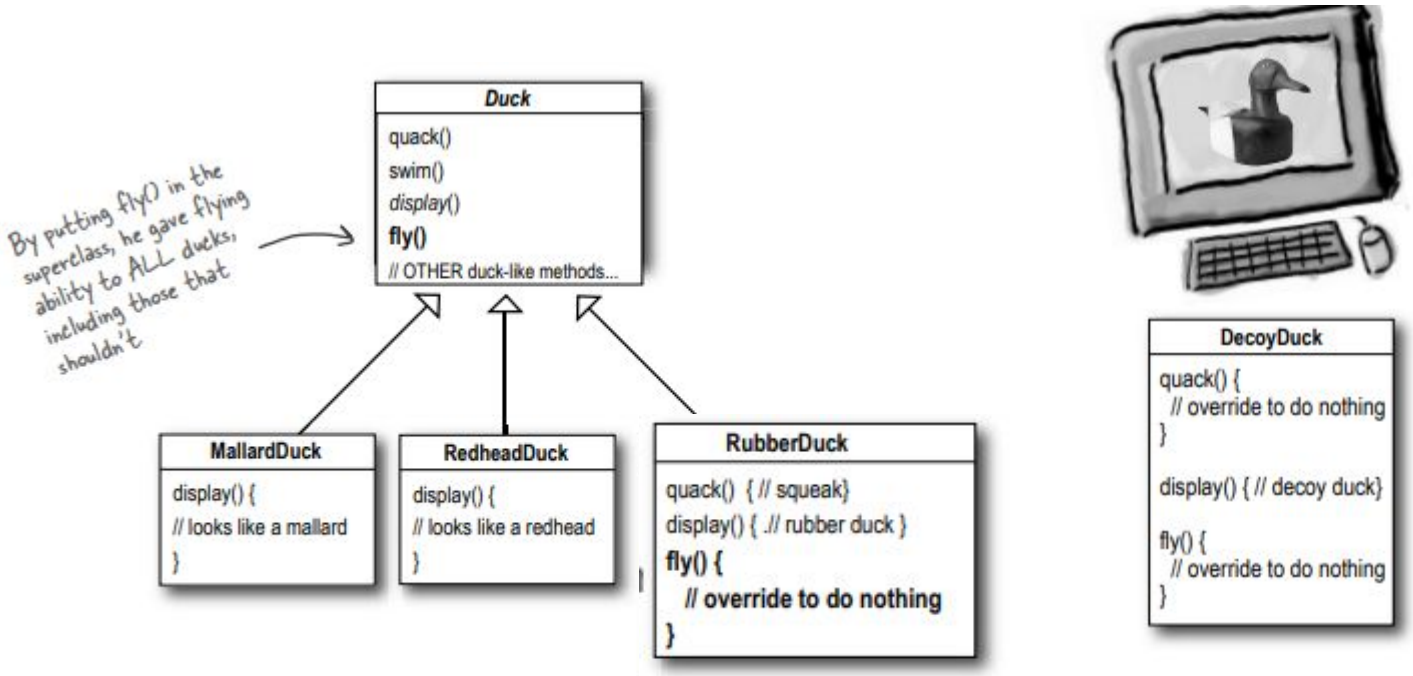


Can fly



Cannot fly

Assume that we have a new decoy duck.



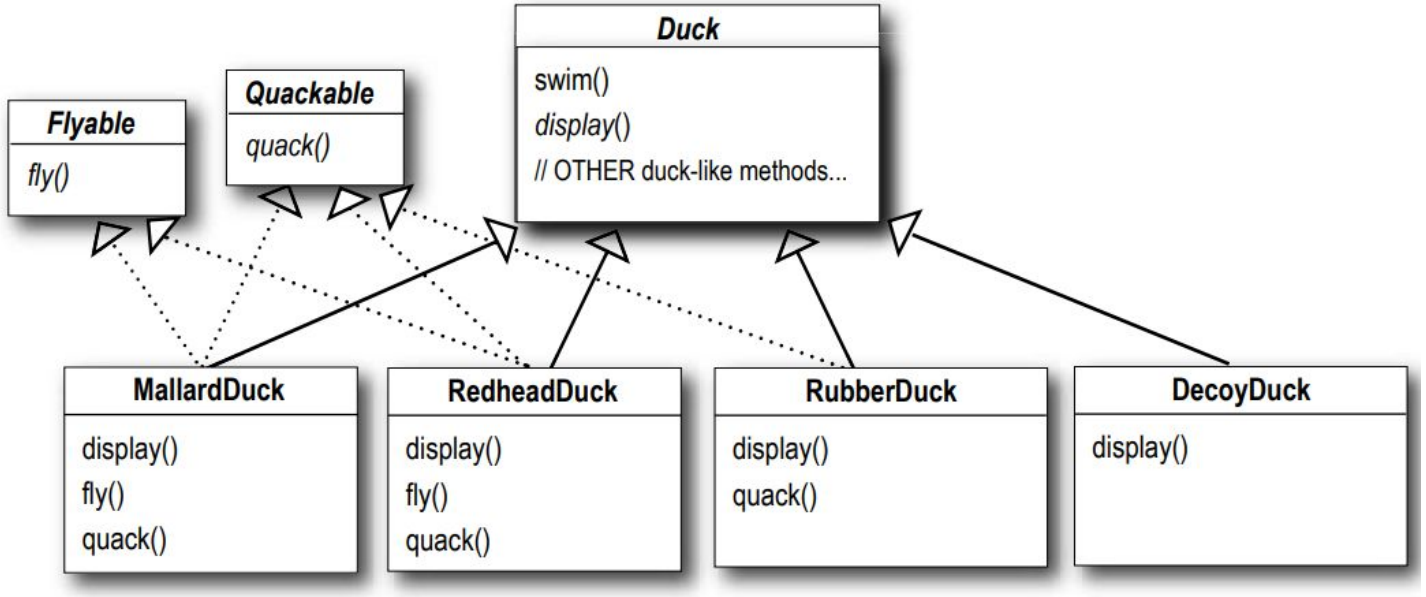
DecoyDuck

```
quack() {
  // override to do nothing
}

display() { // decoy duck}

fly() {
  // override to do nothing
}
```

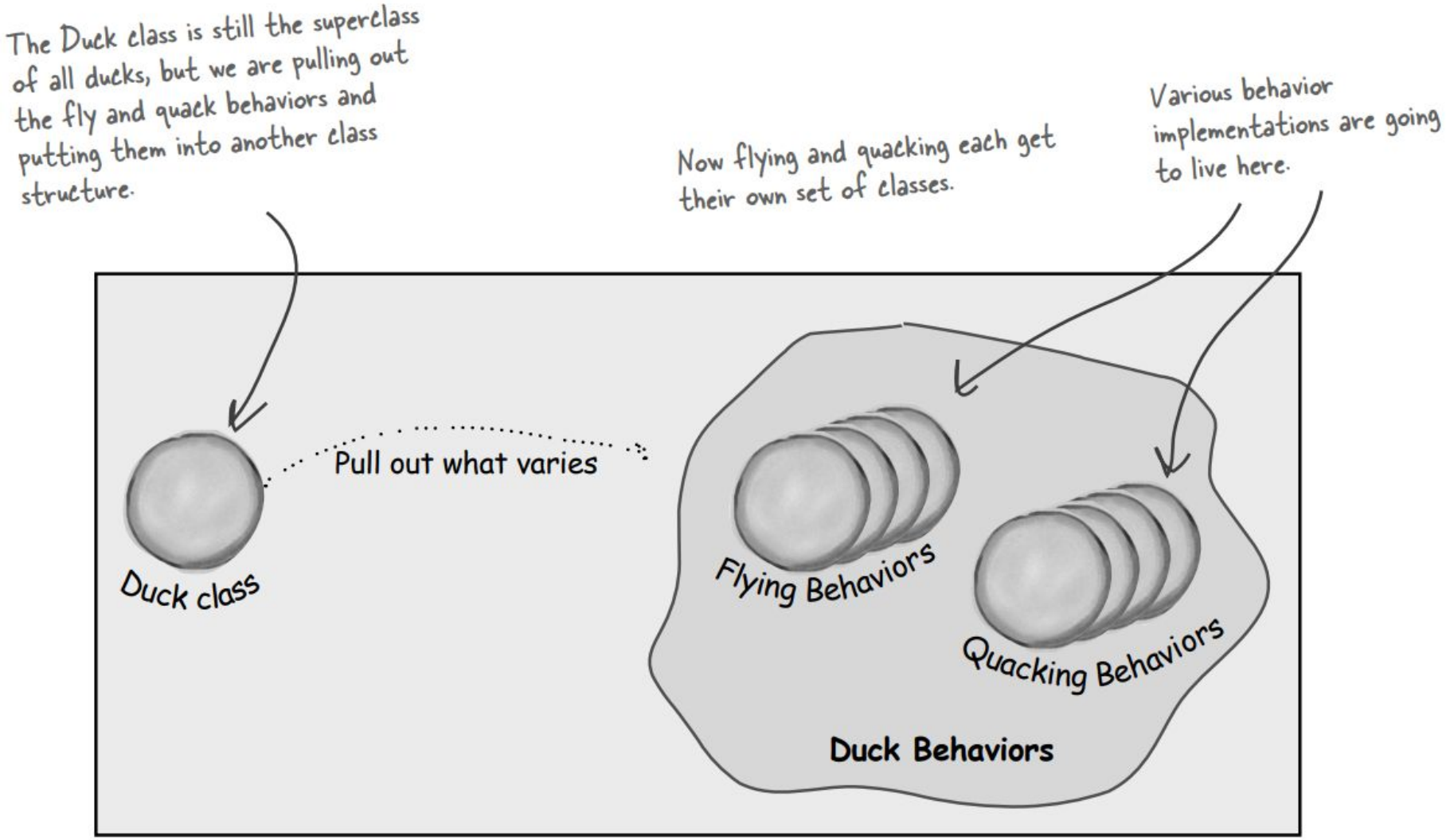
The inheritance probably wasn't the answer

The inheritance probably wasn't the answer, because he just got a memo that says that the executives now want to update the product every six months (in ways they haven't yet decided on).

We know that not all of the subclasses should have flying or quacking behavior, so inheritance isn't the right answer. But while having the subclasses implement Flyable and/or Quackable solves part of the problem (no inappropriately flying rubber ducks), it completely destroys code reuse for those behaviors, so it just creates a different maintenance nightmare.



Pattern Design



- In object-oriented programming (OOP), an interface is a contract that defines a set of methods and properties that a class must implement, without providing any implementation details. It allows different classes to be treated uniformly if they implement the same interface, promoting a form of polymorphism.
- Key features of interfaces include:

  1. Method Signatures: An interface specifies the methods that implementing classes must have, including their names, return types, and parameters, but not their body.
  2. Multiple Implementations: Different classes can implement the same interface in different ways, allowing for flexibility and extensibility.
  3. No State: Interfaces do not maintain state or data; they only define behavior.
  4. Supports Multiple Inheritance: A class can implement multiple interfaces, which helps avoid some limitations of single inheritance in OOP.

  Interfaces are commonly used in design patterns and frameworks to allow for interchangeable components.

What's a design pattern?

- **Design patterns** are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.
- You can’t just find a pattern and copy it into your program, the way you can with off-the-shelf functions or libraries. The pattern is not a specific piece of code, but a general concept for solving a particular problem. You can follow the pattern details and implement a solution that suits the realities of your own program.
- Patterns are often confused with algorithms, because both concepts describe typical solutions to some known problems. While an algorithm always defines a clear set of actions that can achieve some goal, a pattern is a more high-level description of a solution. The code of the same pattern applied to two different programs may be different.
- An analogy to an algorithm is a cooking recipe: both have clear steps to achieve a goal. On the other hand, a pattern is more like a blueprint: you can see what the result and its features are, but the exact order of implementation is up to you.

Three types of patterns

- **Creational patterns** provide object creation mechanisms that increase flexibility and reuse of existing code.
- **Structural patterns** explain how to assemble objects and classes into larger structures, while keeping these structures flexible and efficient.
- **Behavioral patterns** take care of effective communication and the assignment of responsibilities between objects

Behavioral Design Patterns

Chain of Responsibility

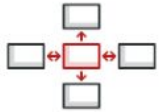
Lets you pass requests along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.

Command

Turns a request into a stand-alone object that contains all information about the request. This transformation lets you pass requests as a method arguments, delay or queue a request's execution, and support undoable operations.

Iterator

Lets you traverse elements of a collection without exposing its underlying representation (list, stack, tree, etc.).



Mediator

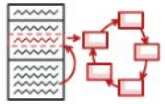
Lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object.

Memento

Lets you save and restore the previous state of an object without revealing the details of its implementation.

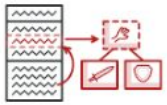
Observer

Lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing.



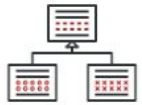
State

Lets an object alter its behavior when its internal state changes. It appears as if the object changed its class.



Strategy

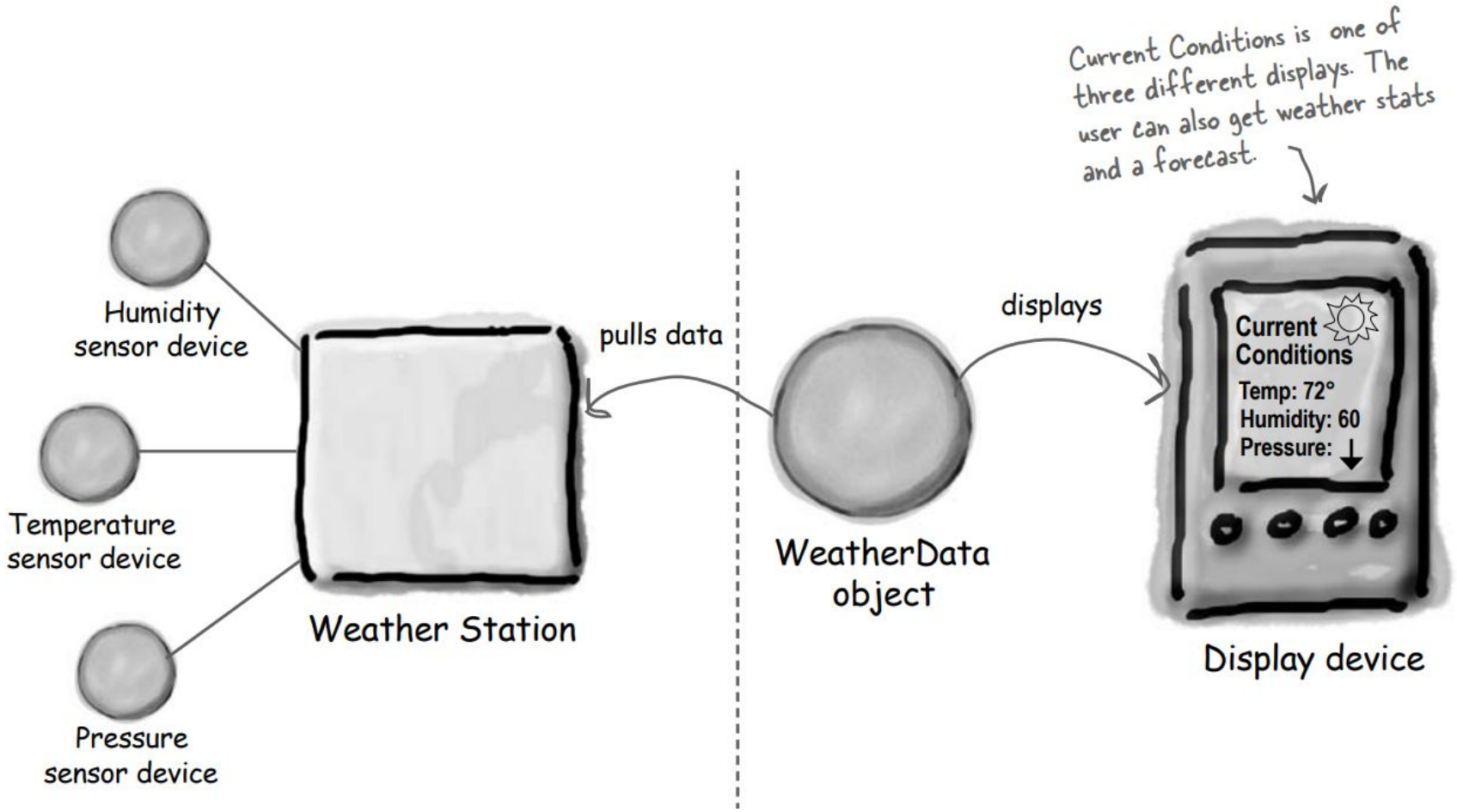
Lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable.



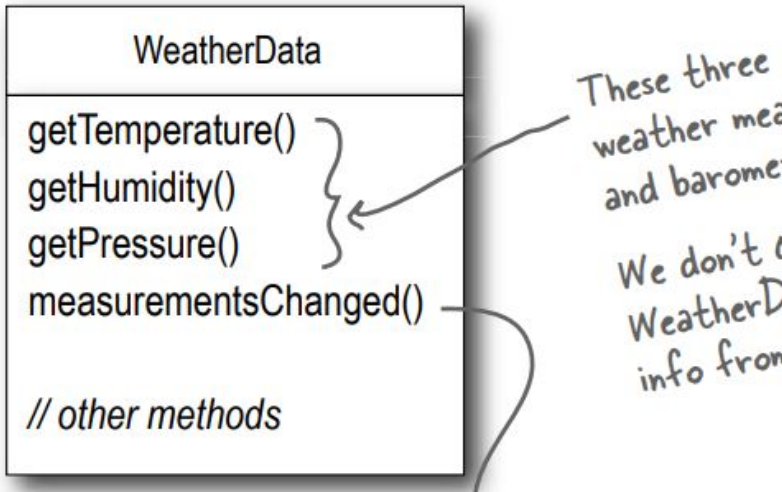
Template Method

Defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.

Observer Pattern



Implementation



These three methods return the most recent weather measurements for temperature, humidity and barometric pressure respectively.

We don't care HOW these variables are set; the WeatherData object knows how to get updated info from the Weather Station.

The developers of the WeatherData object left us a clue about what we need to add...

```
/*
 * This method gets called
 * whenever the weather measurements
 * have been updated
 *
 */
public void measurementsChanged() {
    // Your code goes here
}
```

WeatherData.java

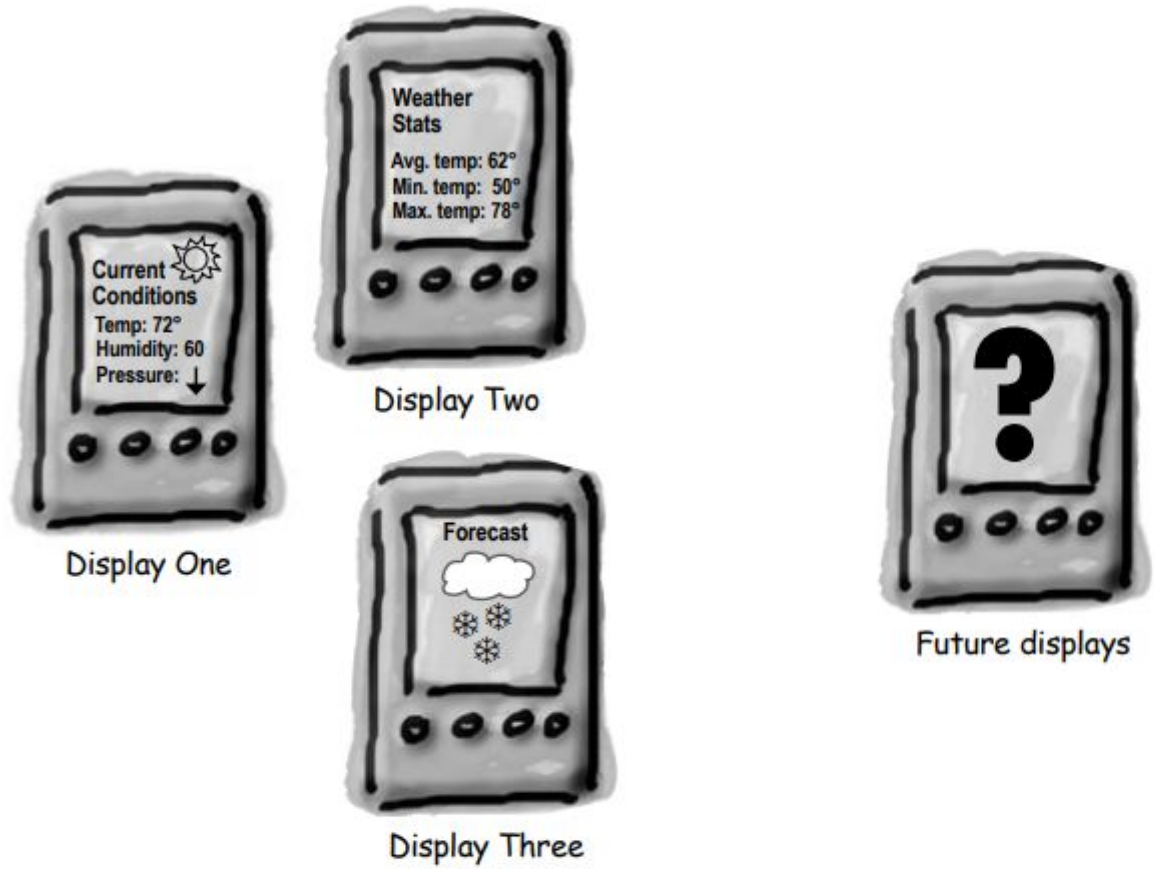
Our Problem

Remember, this Current Conditions is just ONE of three different display screens.

Display device

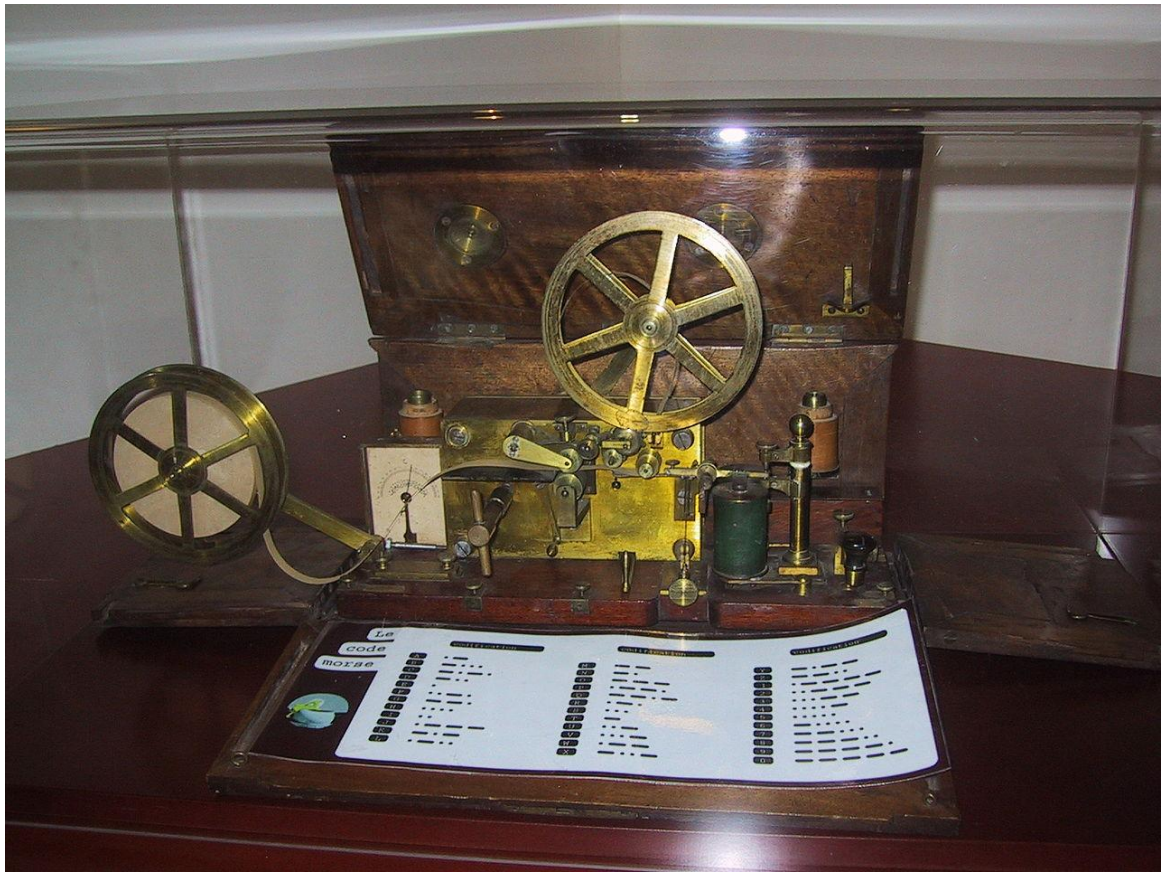
Our job is to implement measurementsChanged() so that it updates the three displays for current conditions, weather stats, and forecast.

```
getTemperature()
getHumidity()
getPressure()


measurementsChanged()
```



Implementatiom

Telegraph



Developed in the 1830s and 1840s by Samuel Morse (1791-1872) and other inventors, the telegraph revolutionized long-distance communication. It worked by transmitting electrical signals over a wire laid between stations.

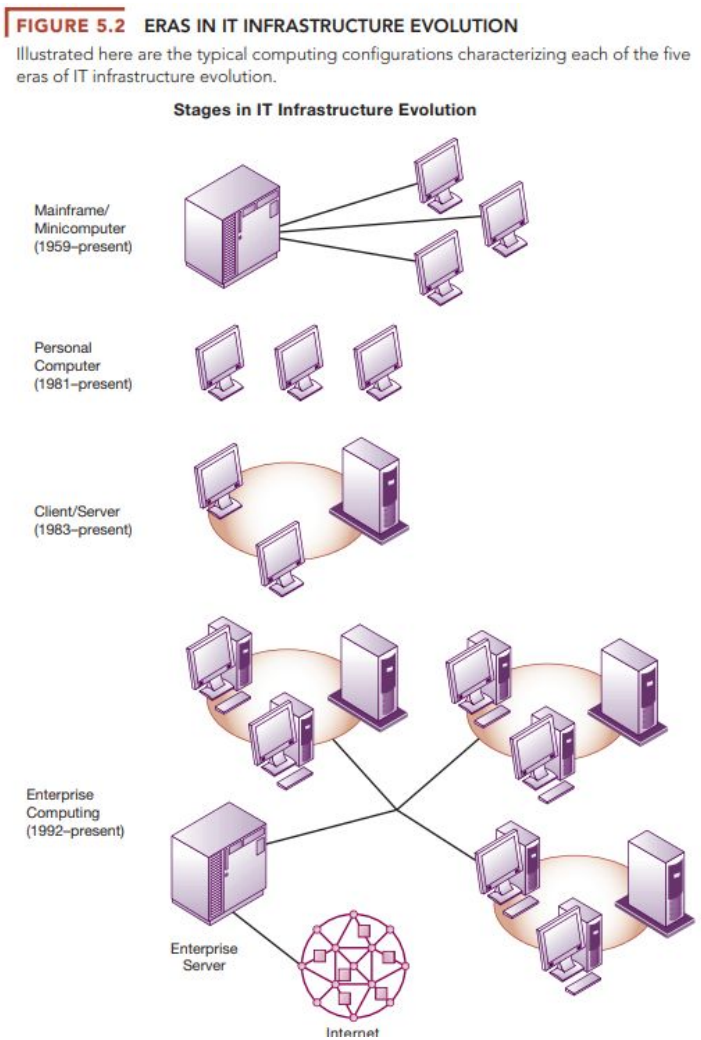
Fax

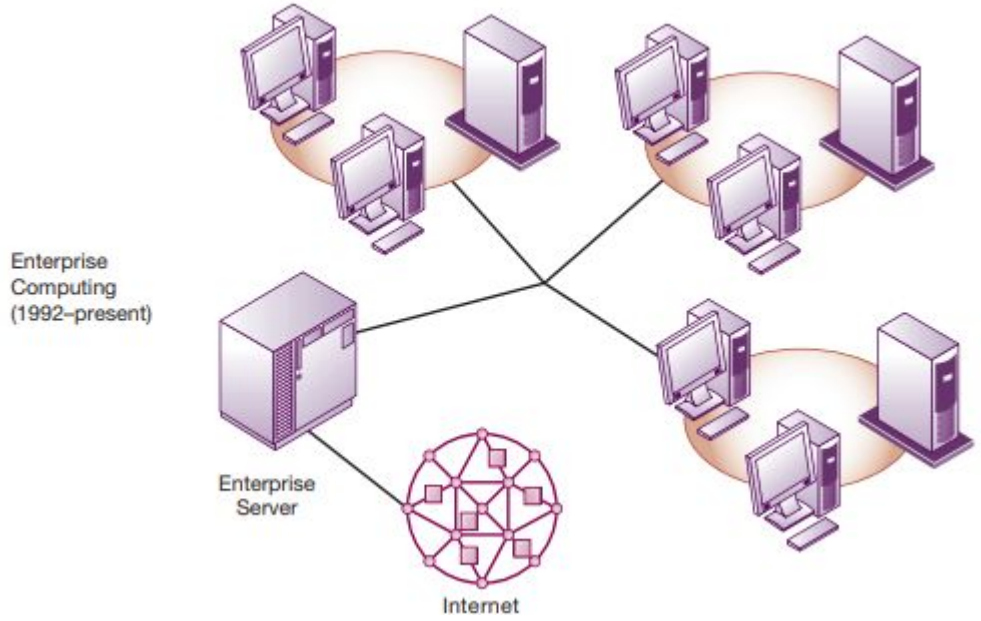
- Can send images and papers

Client/server (1983–now)

- Client–server systems began to emerge in the United States in the early 1980s as computing transitioned from large mainframes to distributed processing using multiple workstations or personal computers. Corporations quickly adopted client–server systems, which became the backbones of their office automation and communication infrastructure.

Network in any organization





The curent network

FIGURE 7.1 COMPONENTS OF A SIMPLE COMPUTER NETWORK

Illustrated here is a simple computer network consisting of computers, a network operating system (NOS) residing on a dedicated server computer, cable (wiring) connecting the devices, switches, and a router.

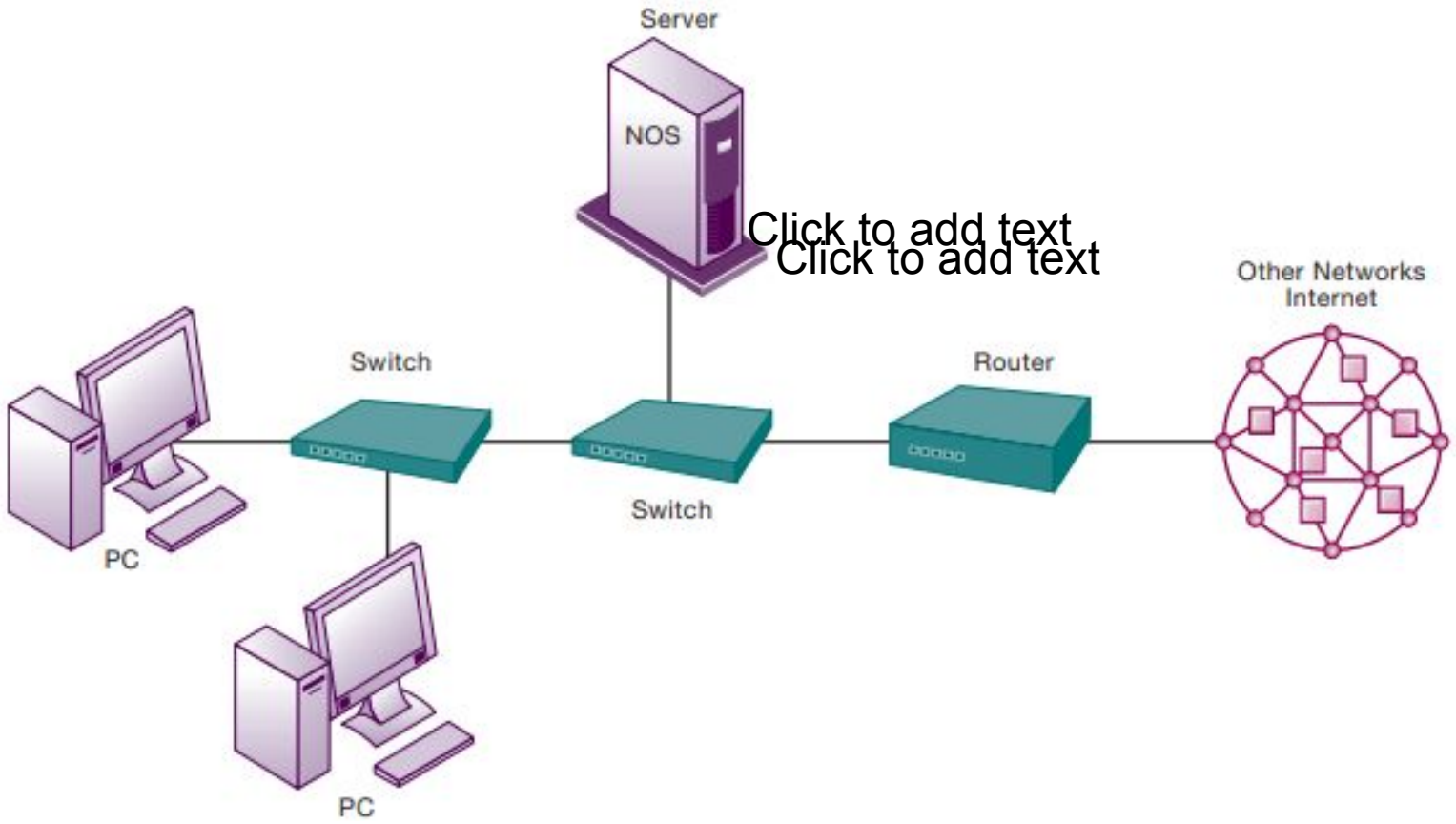
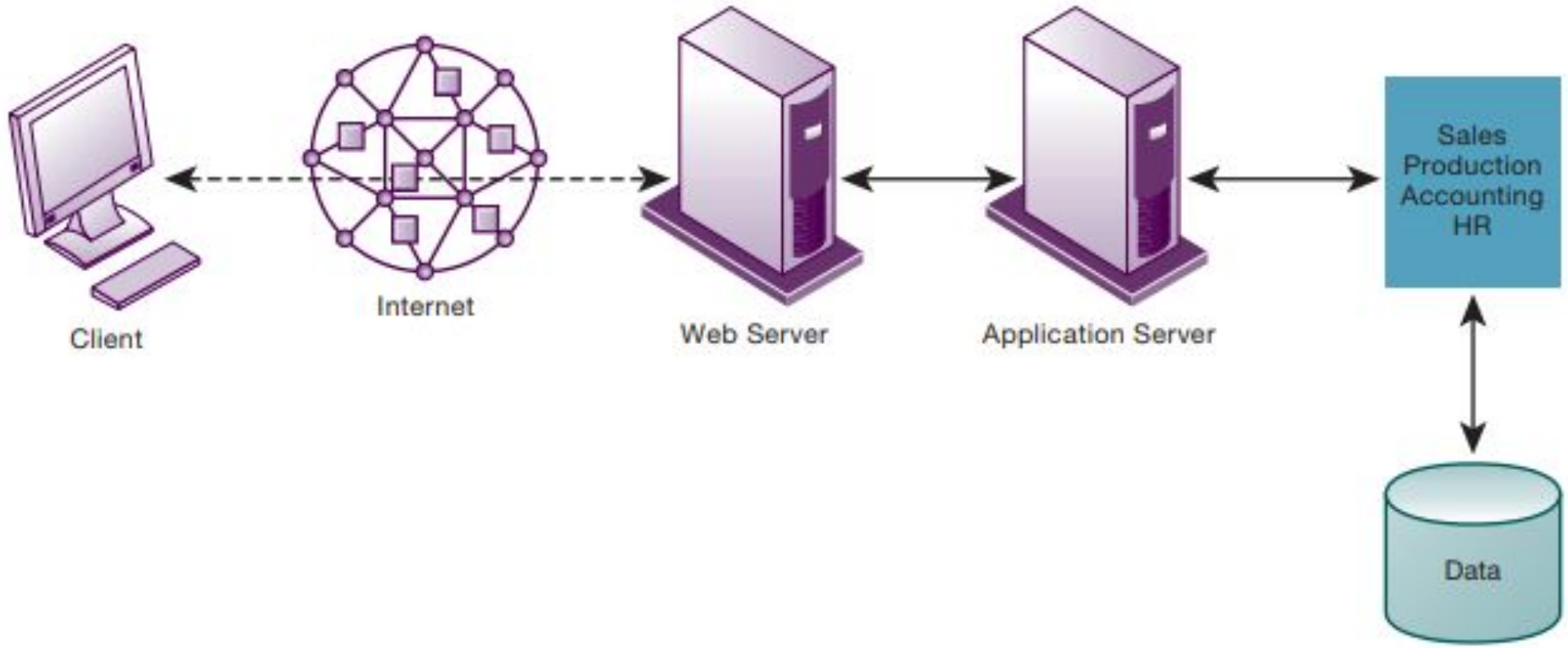
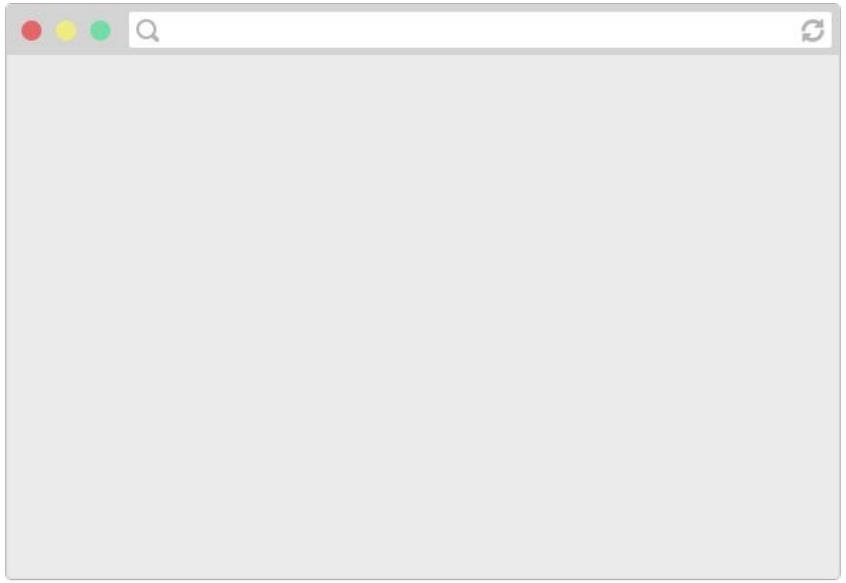


FIGURE 5.3 A MULTITIERED (N-TIER) CLIENT/SERVER NETWORK

In a multitiered client/server network, client requests for service are handled by different levels of servers.

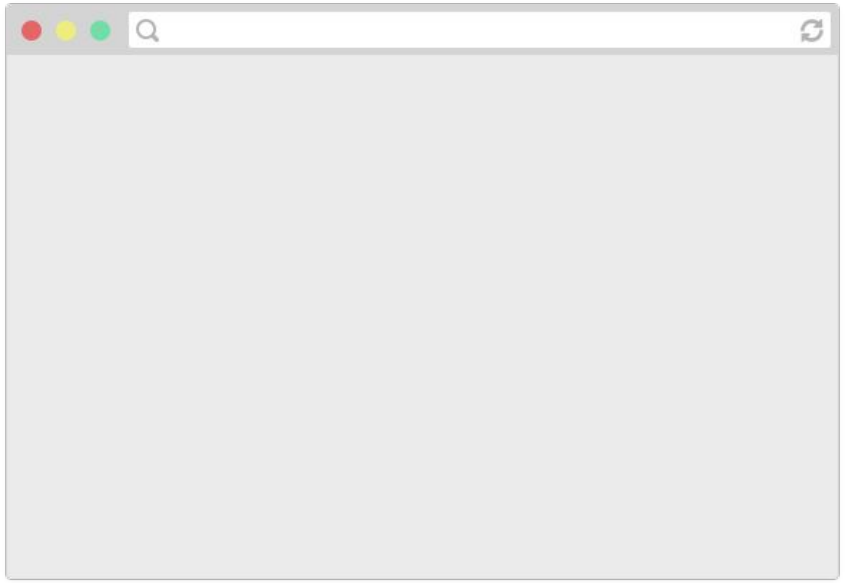


How do web pages work?



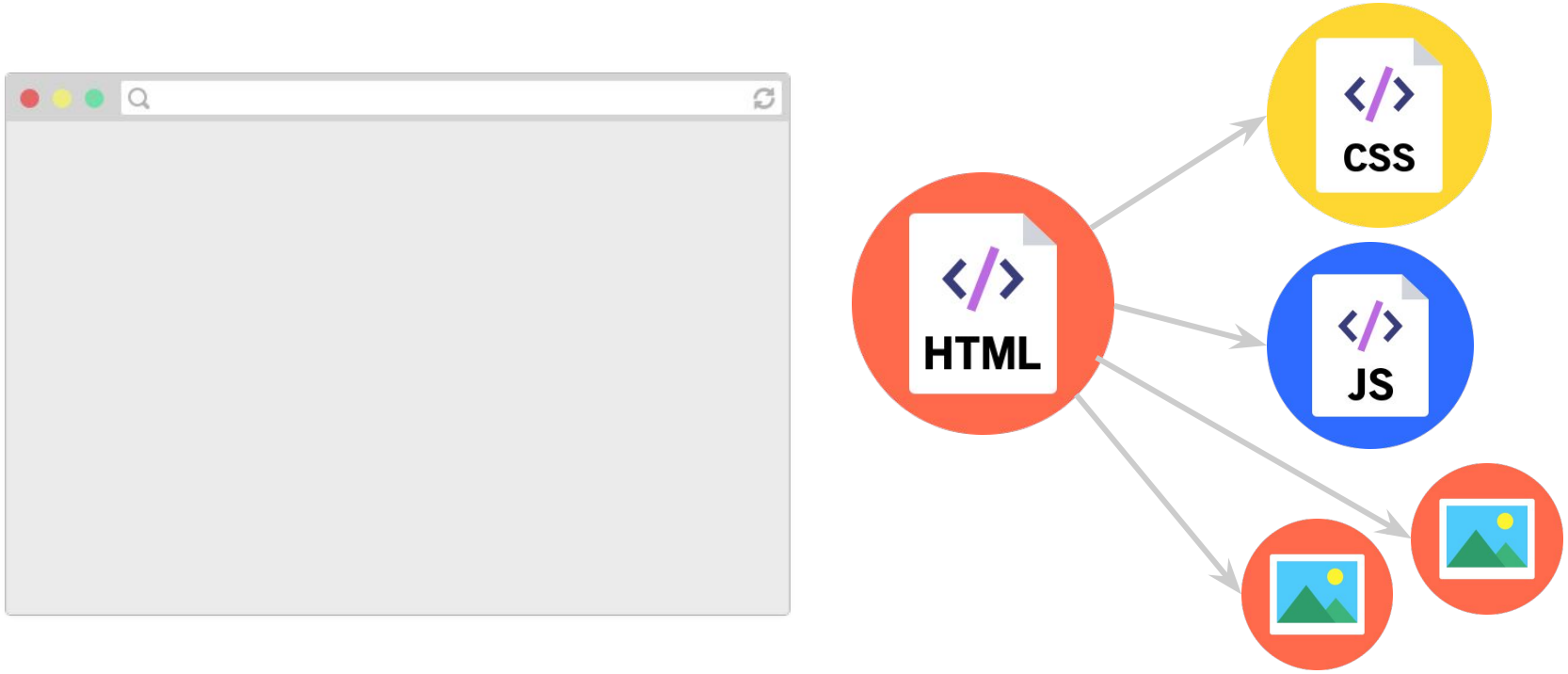
Browsers are applications that can display web pages.
E.g. Chrome, Firefox, Safari, Internet Explorer, Edge, etc.

How do web pages work?



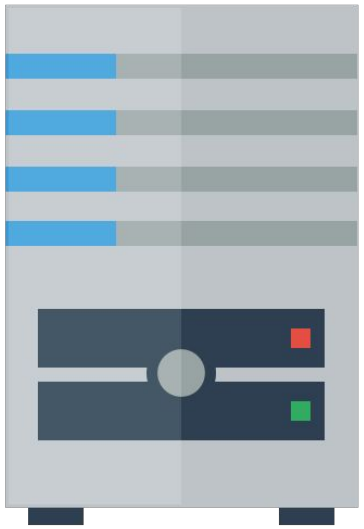
Web pages are written in a markup language called **HTML**, so browsers display a web page by reading and interpreting its HTML.

How do web pages work?



The HTML file might link to other resources, like images, videos, as well as **JavaScript** and **CSS** (stylesheet) files, which the browser then also loads.

How do web pages work?

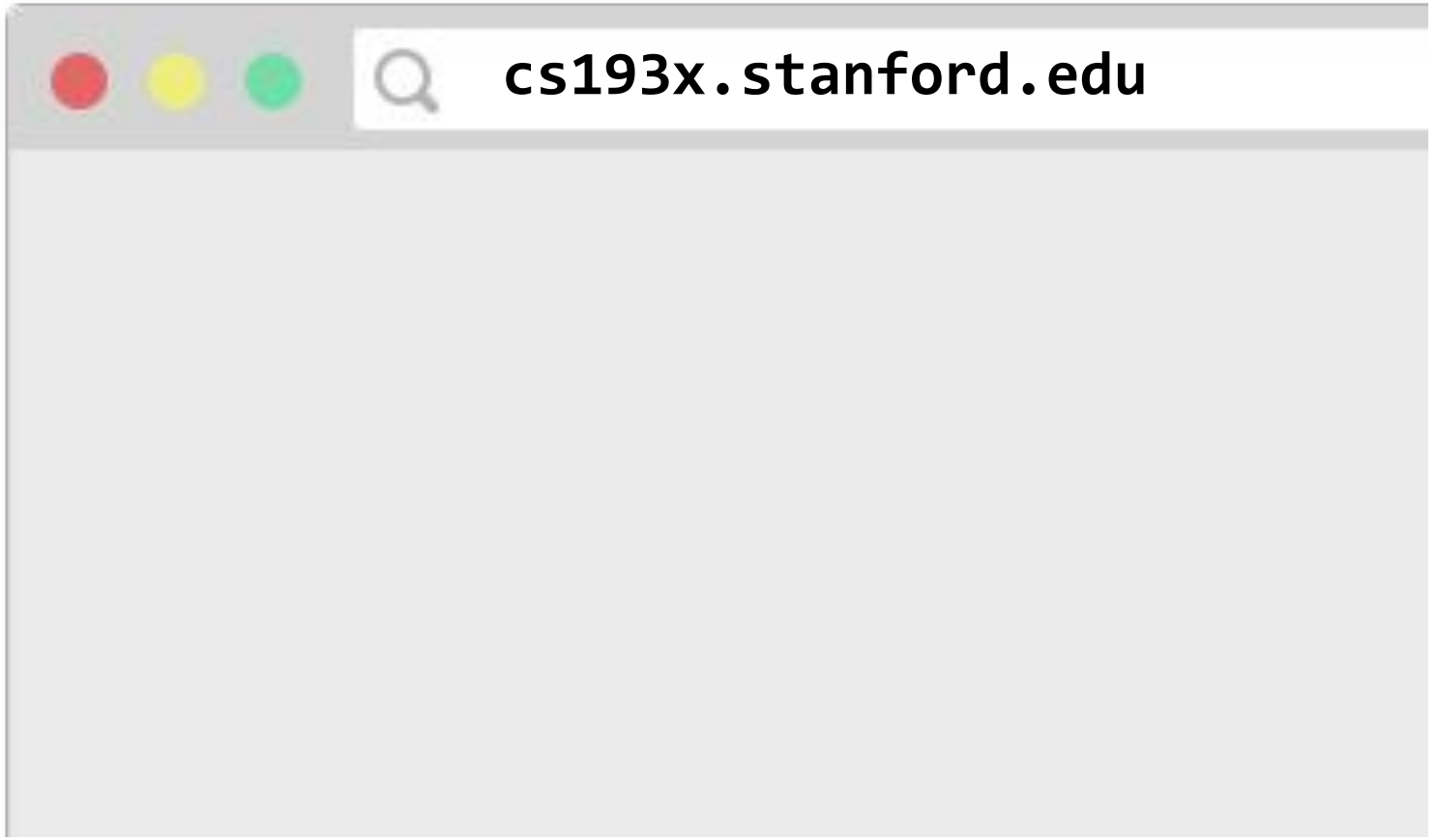


A **web server** is a program running on a computer that delivers web pages in response to requests.

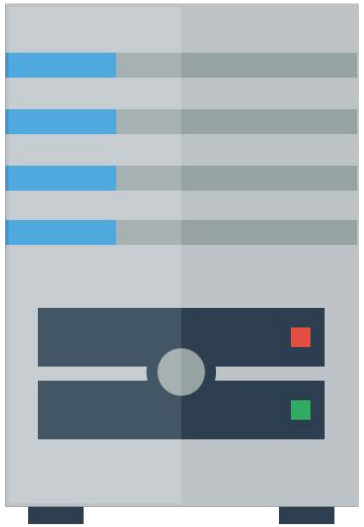
It either stores or generates the web page returned.

How do web pages work?

1. You type in a URL, which is the address of the HTML file on the internet.

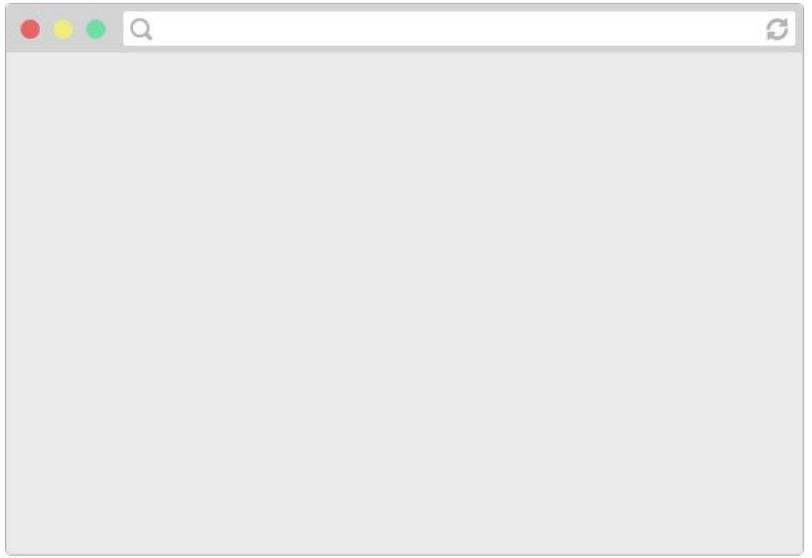


How do web pages work?

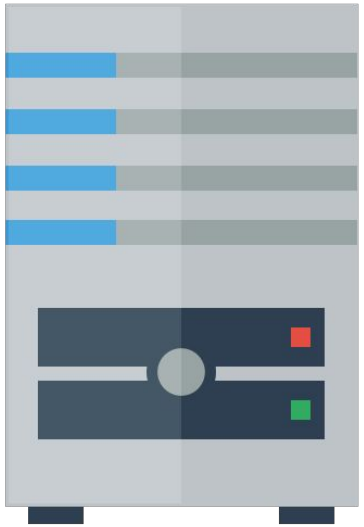


2. The browser asks the web server that hosts the document to send that document.

GET

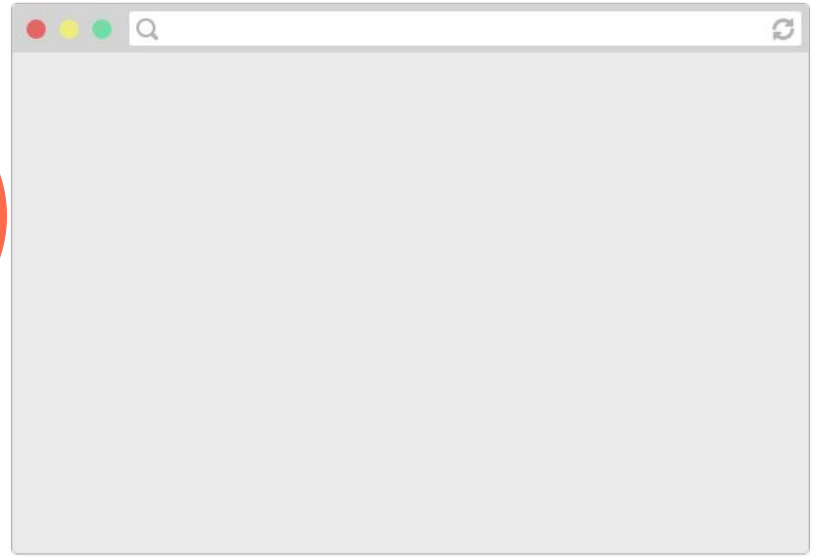


How do web pages work?



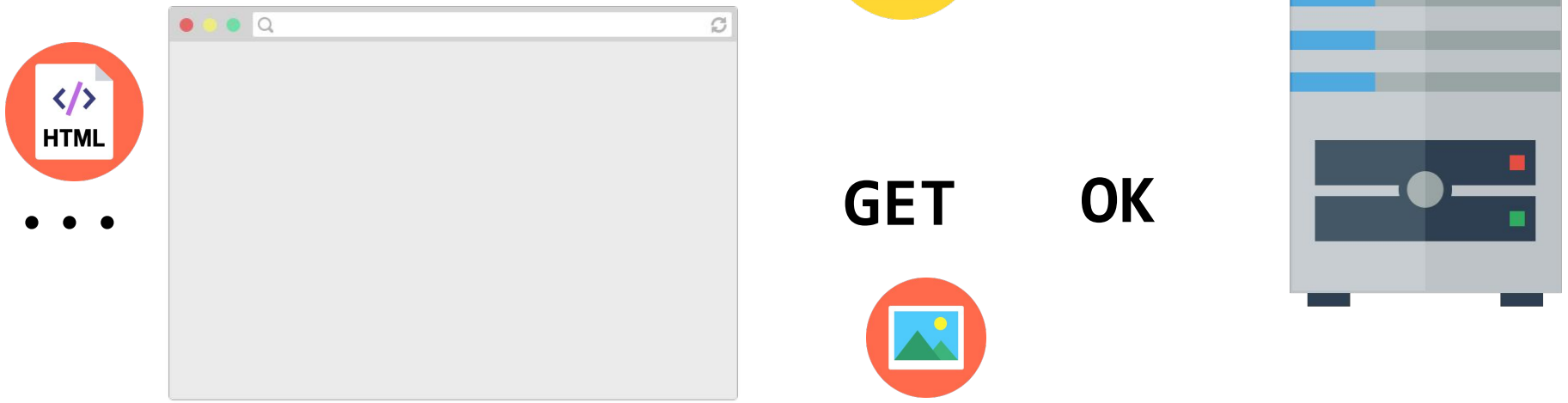
3. The web server responds to the browser with HTML file that was requested.

OK



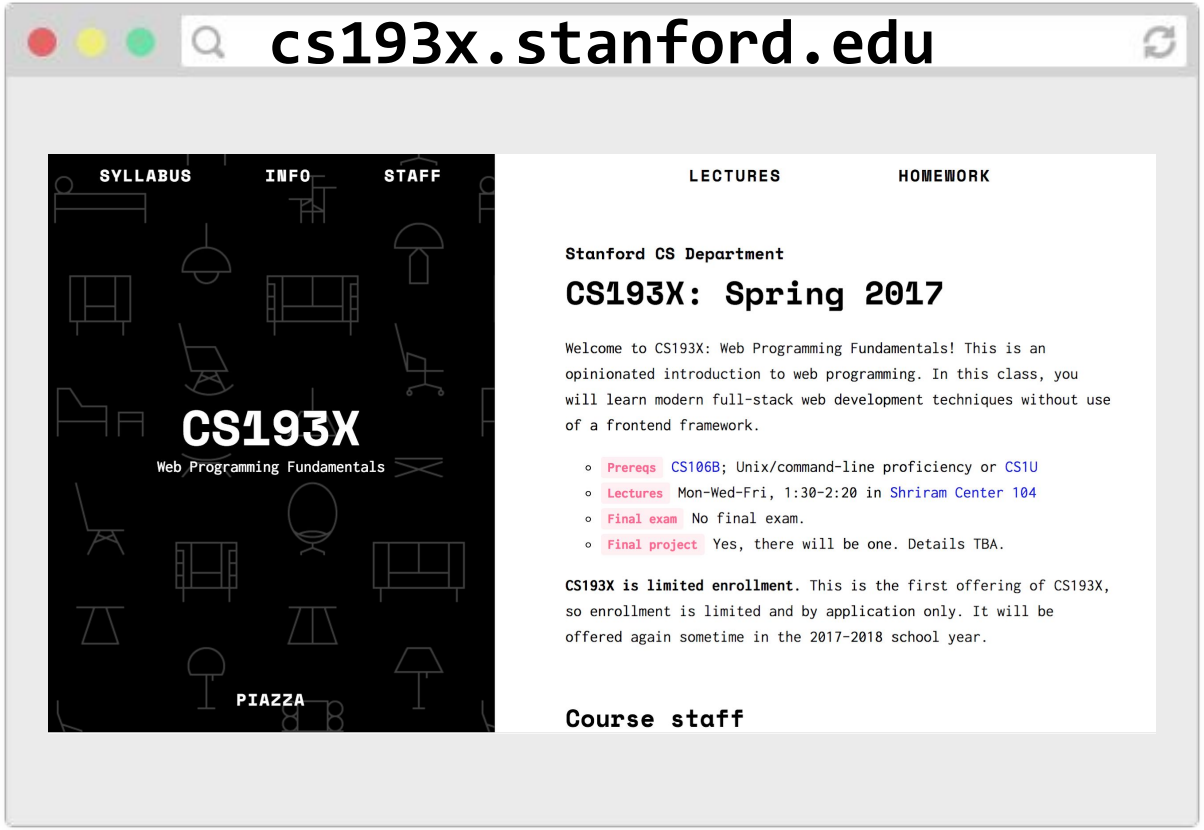
How do web pages work?

4. The browser reads the HTML, sees the embedded resources and asks the server for those as well.

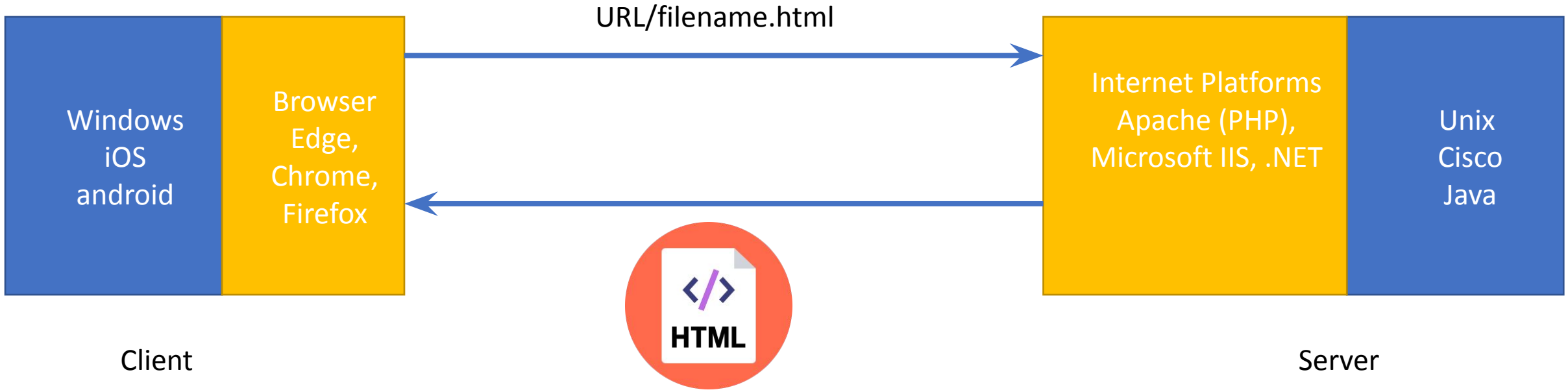


How do web pages work?

5. The web page is loaded when all the resources are fetched and displayed.



What is the main problem in this client/server platform?



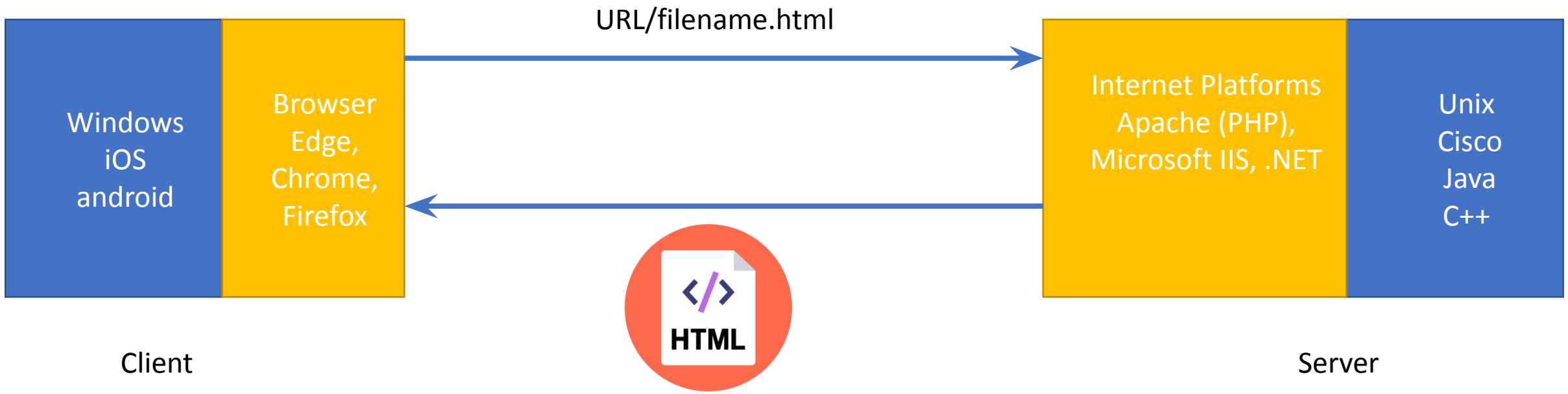
HTML is static, no dynamics

Amazon.com1995 July 16th

Link and information

Common-gateway interface

The client can run c++/java program in the client. This program generates outputs as HTML format



The main problem, is we need proffessional programmers

HTML is static, no dynamics

Active-pages

- PHP, ASP, .NET, JSP

Server side

```
The server's current time:
<%
Response.Write Now()
%>
```

Client Side

```
The server's current time:
8/11/2015 6:24:45 PM
```

insert_data.php × ls_one.php

sql_basics > insert_data.php

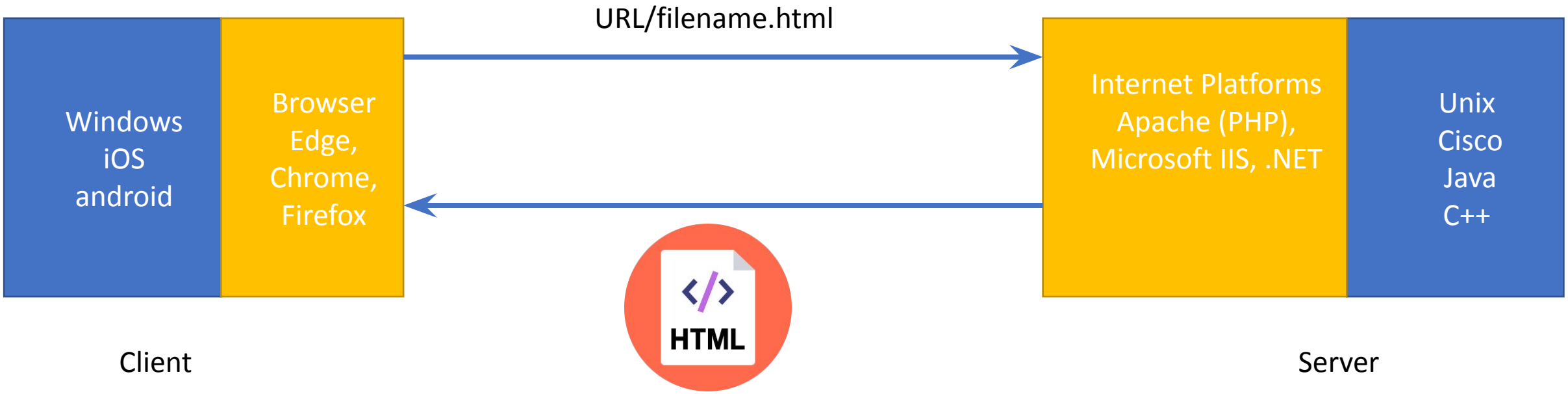
```php
<?php

    //Create Connection
    //mysqli_connect('HOSTNAME','DATABASE USER','DATABASE PASSWORD','DATABASE NAME');
    $conn = mysqli_connect('localhost','root','','myfirstdb');

    //CHECK CONNECTION
    if (!$conn) {
        die("Connection failed: ".mysqli_connect_error());
    }

    $sql = "INSERT INTO users (email, password) VALUES ('c@c.com', 'hello')";
    $conn->query($sql);

    echo "New User Added";
?>
```

Table Column Column Value Value

Easy to connect to Data Management and Storage
IBM DB2 Oracle SQL Server
Sybase
MySQL
Apache Hadoop

Progress in the client

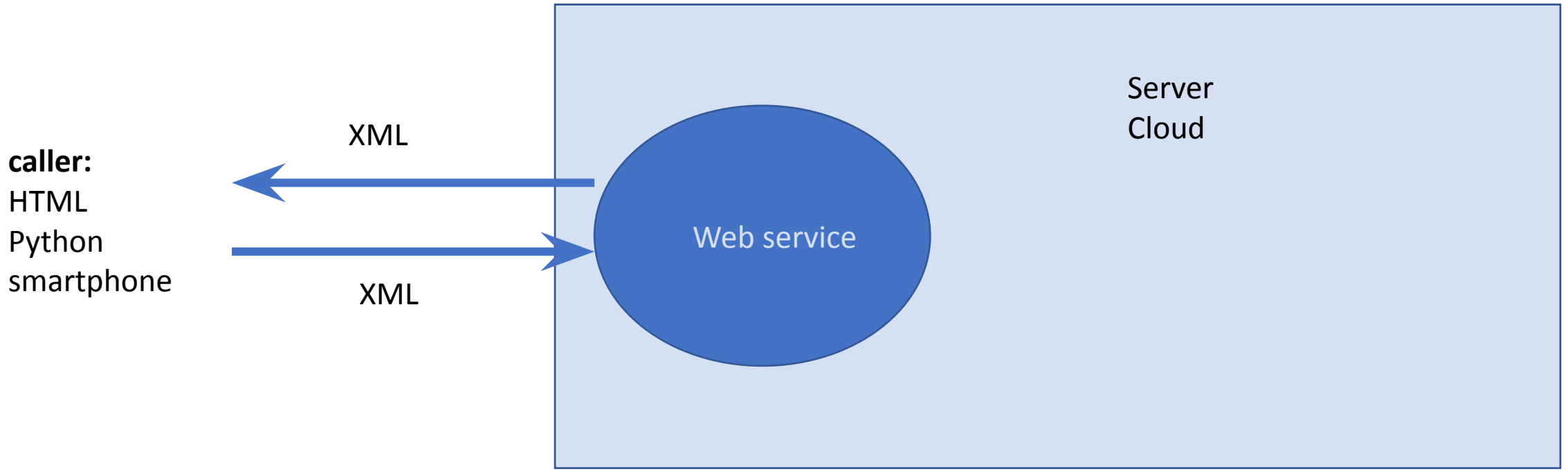
JavaScript, programming in client
Jquery, AJAX, Angular, ..., etc



HTML is static, no dynamics

The current technology: Web service/microservice

Web services refer to a set of loosely coupled software components that exchange information with each other using universal web communication standards and languages. They can exchange information between two different systems regardless of the operating systems or programming languages on which the systems are based. They can be used to build open standard web-based applications linking systems of two different organizations, and they can also be used to create applications that link disparate systems within a single company. Different applications can use web services to communicate with each other in a standard way without time-consuming custom coding.



- The collection of web services that are used to build a firm's software systems constitutes what is known as a service-oriented architecture**. A service oriented architecture (SOA) is set of self-contained services that communicate with each other to create a working software application.** Business tasks are accomplished by executing a series of these services. Software developers reuse these services in other combinations to assemble other applications as needed.
  - Large companies can have 1000s of microservices

FIGURE 5.10 AMAZON WEB SERVICES

Amazon Web Services (AWS) is a collection of web services that Amazon provides to users of its cloud platform. AWS is the largest provider of cloud computing services in the United States.

Microservice in reality

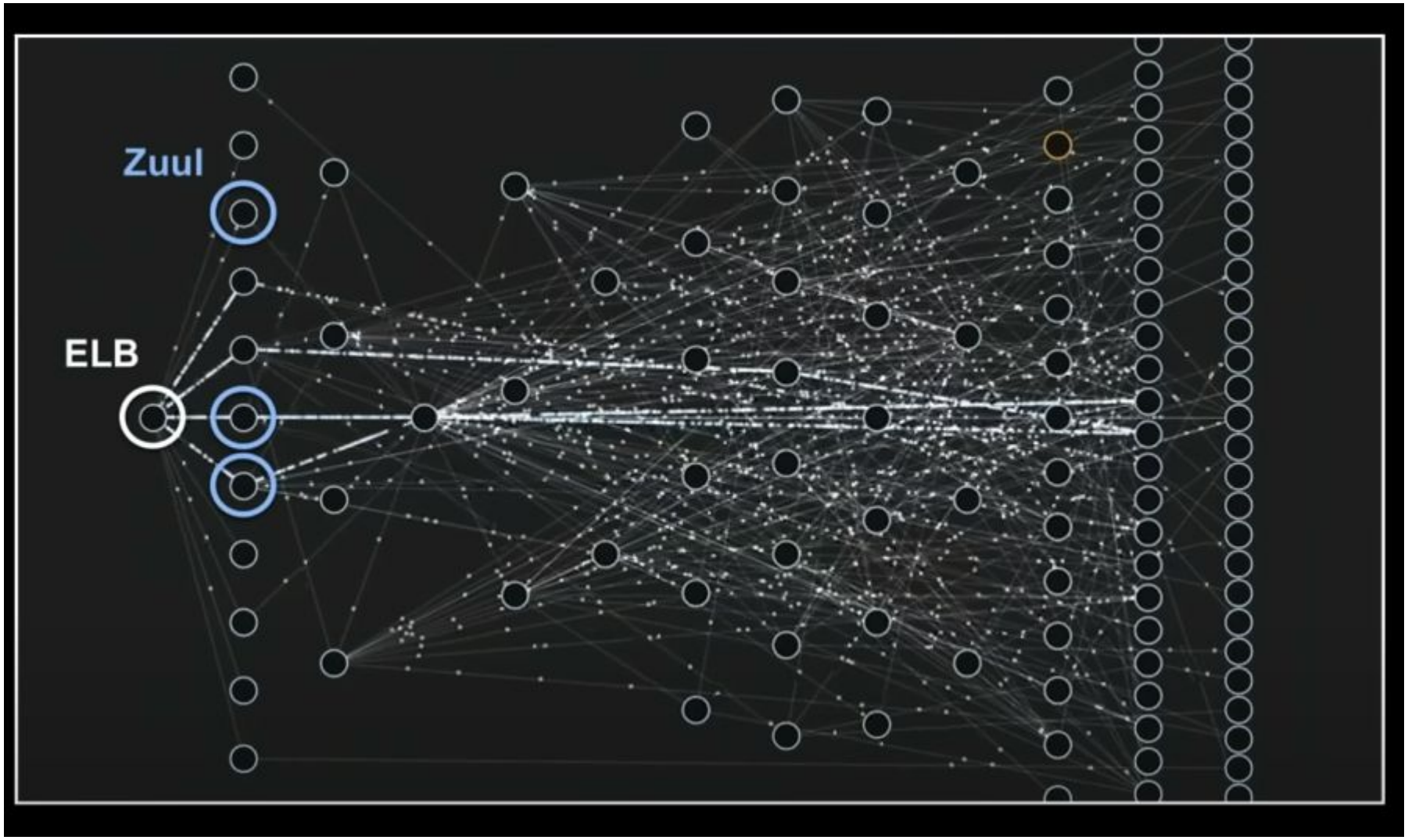
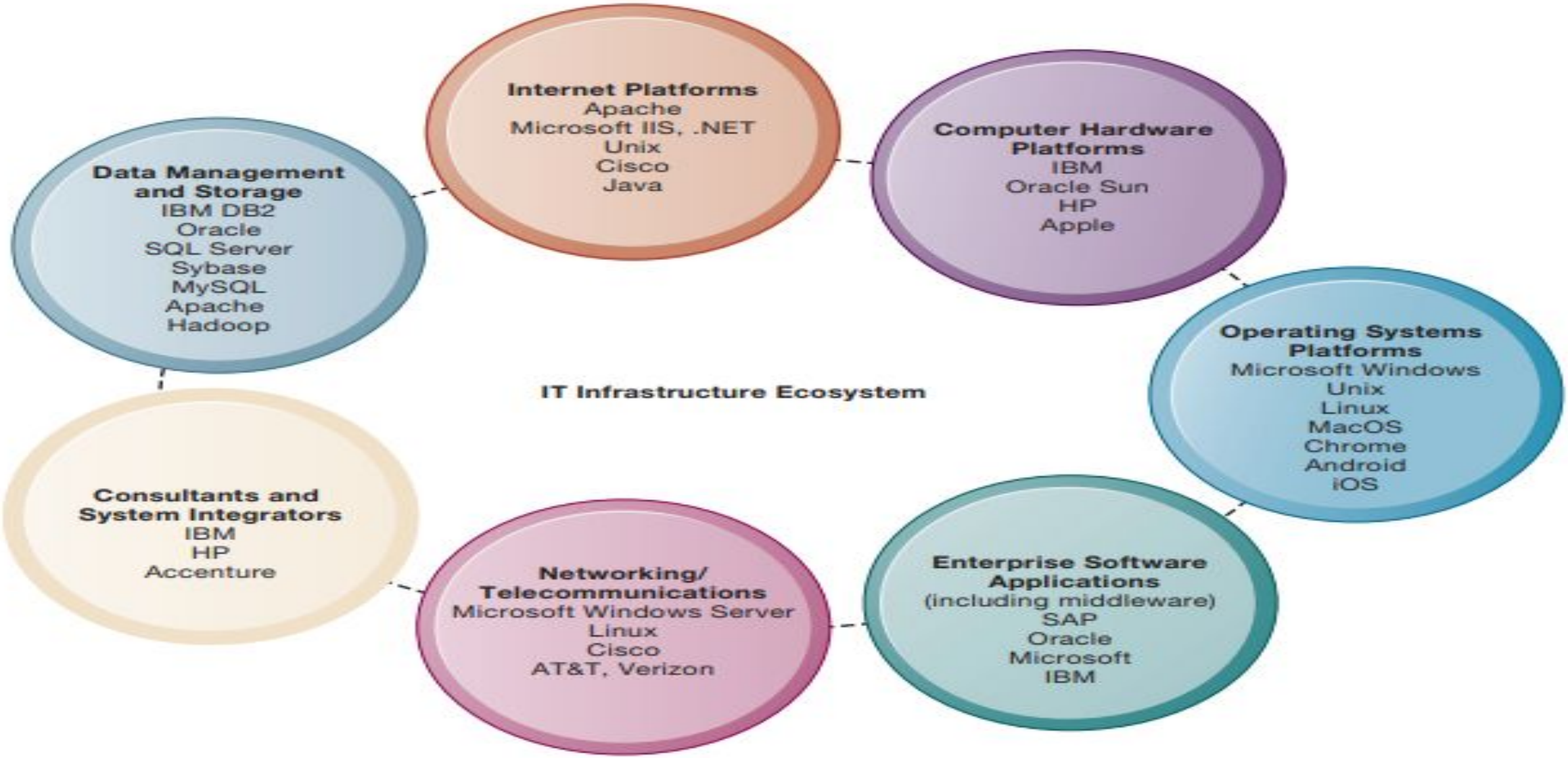


FIGURE 5.8 THE IT INFRASTRUCTURE ECOSYSTEM

There are seven major components that must be coordinated to provide the firm with a coherent IT infrastructure. Listed here are major technologies and suppliers for each component.

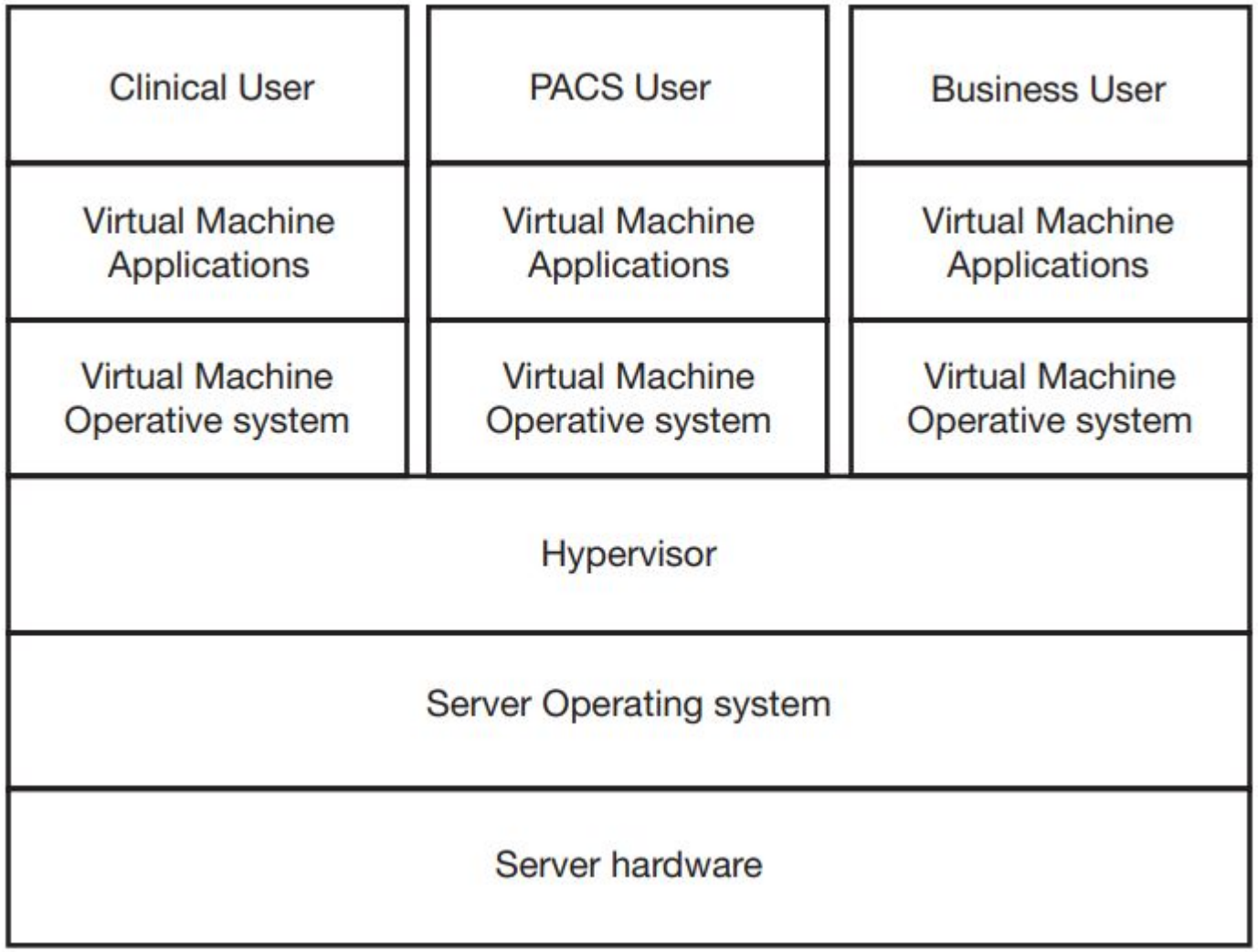


Homework

- summarize this video

https://www.youtube.com/watch?v=CZ3wIuvmHeM

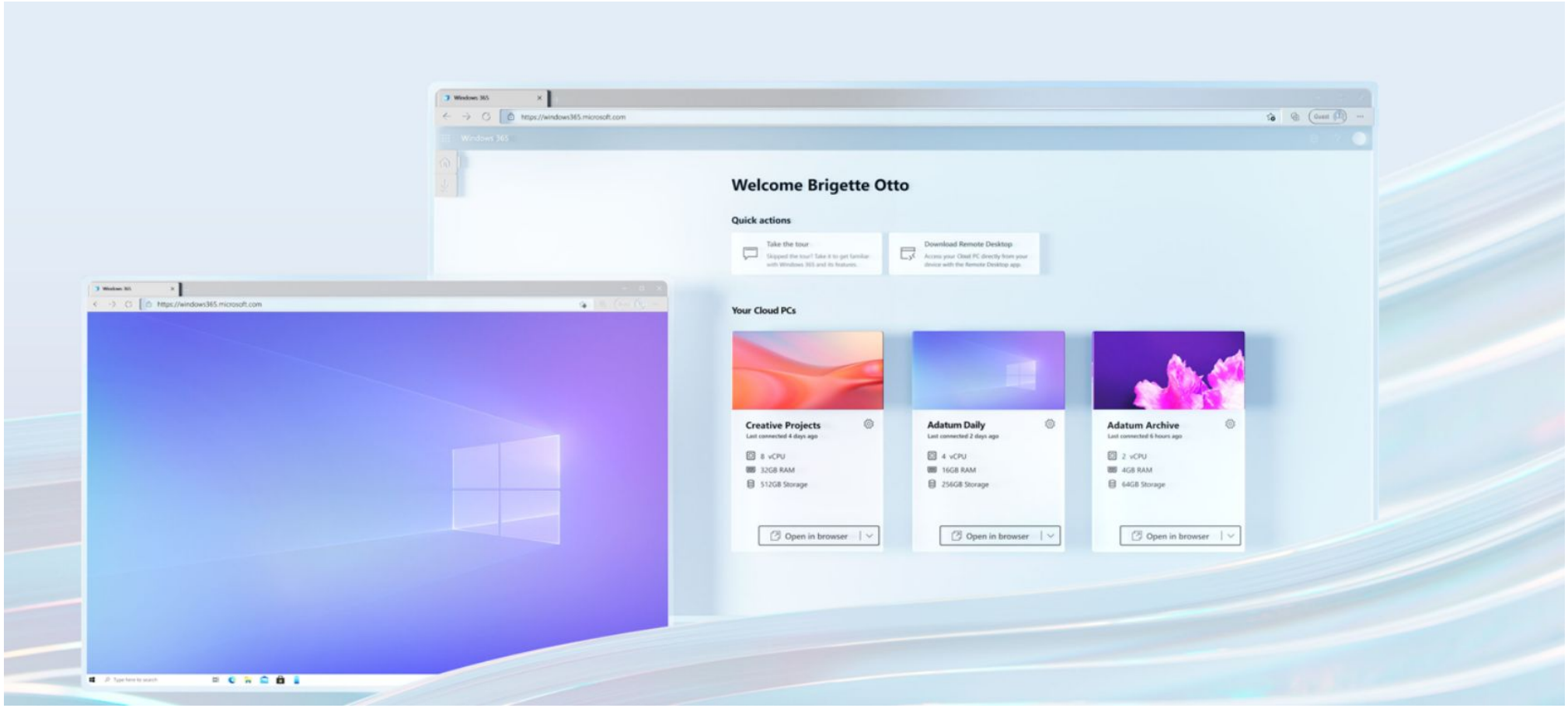
Server Virtualization



Read more VMware

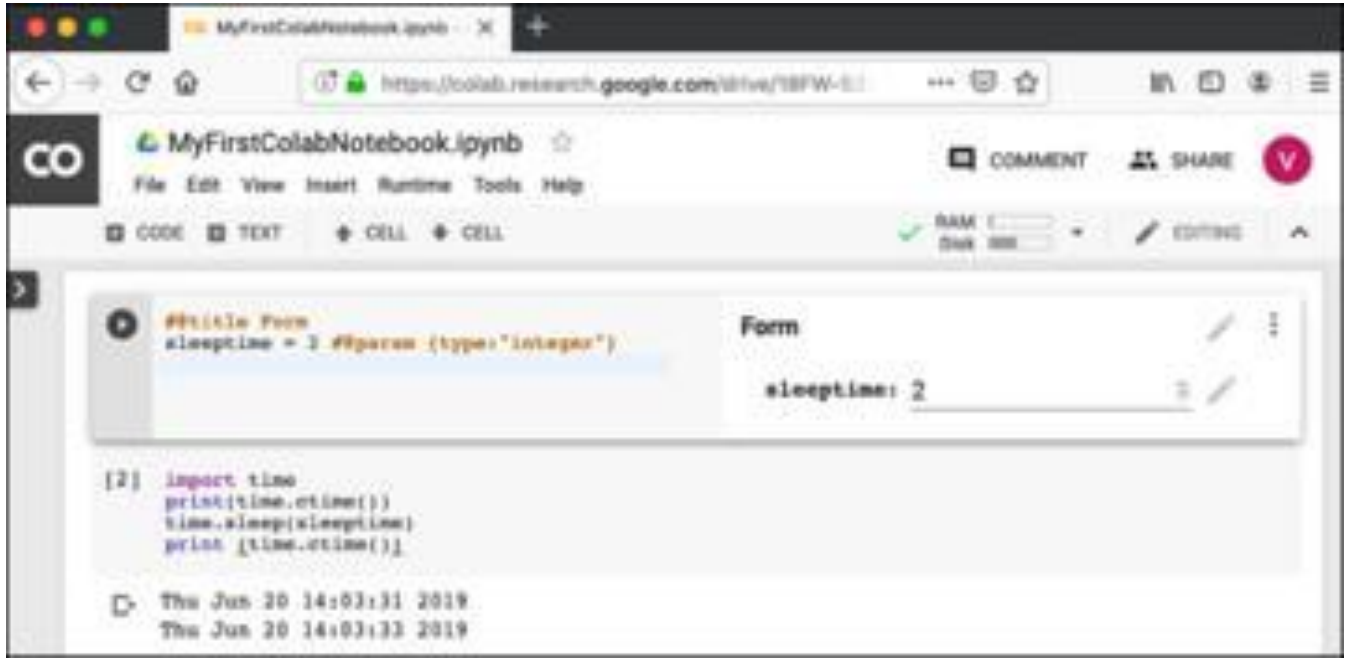
Cloud computing: windows-365

https://www.microsoft.com/en-us/windows-365 (summarize this as a homework)



Cloud computing: google Co-lab

- **You can run your code in google computers**



Cloud computing characteristics

- **Cloud Computing Provides *On-Demand Self-Service***
- **Cloud Computing Encompasses the Characteristics of Grid Computing**
- **Cloud Computing Encompasses the Characteristics of Utility Computing**
- **Cloud Computing Uses Broad Network Access**

  The cloud provider's computing resources are available over a network, accessed with a web browser, and they are configured so that they can be used with any computing device.

Cloud Computing Pools Computing Resources

Cloud Computing Often Occurs on Virtualized Servers

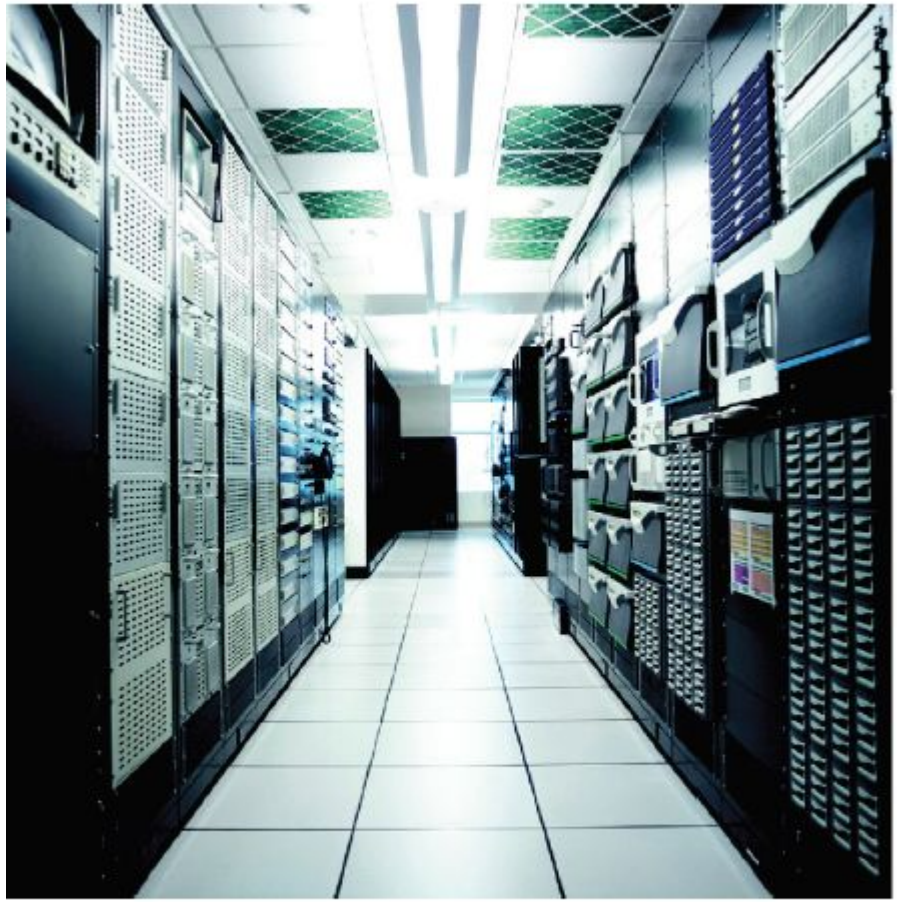
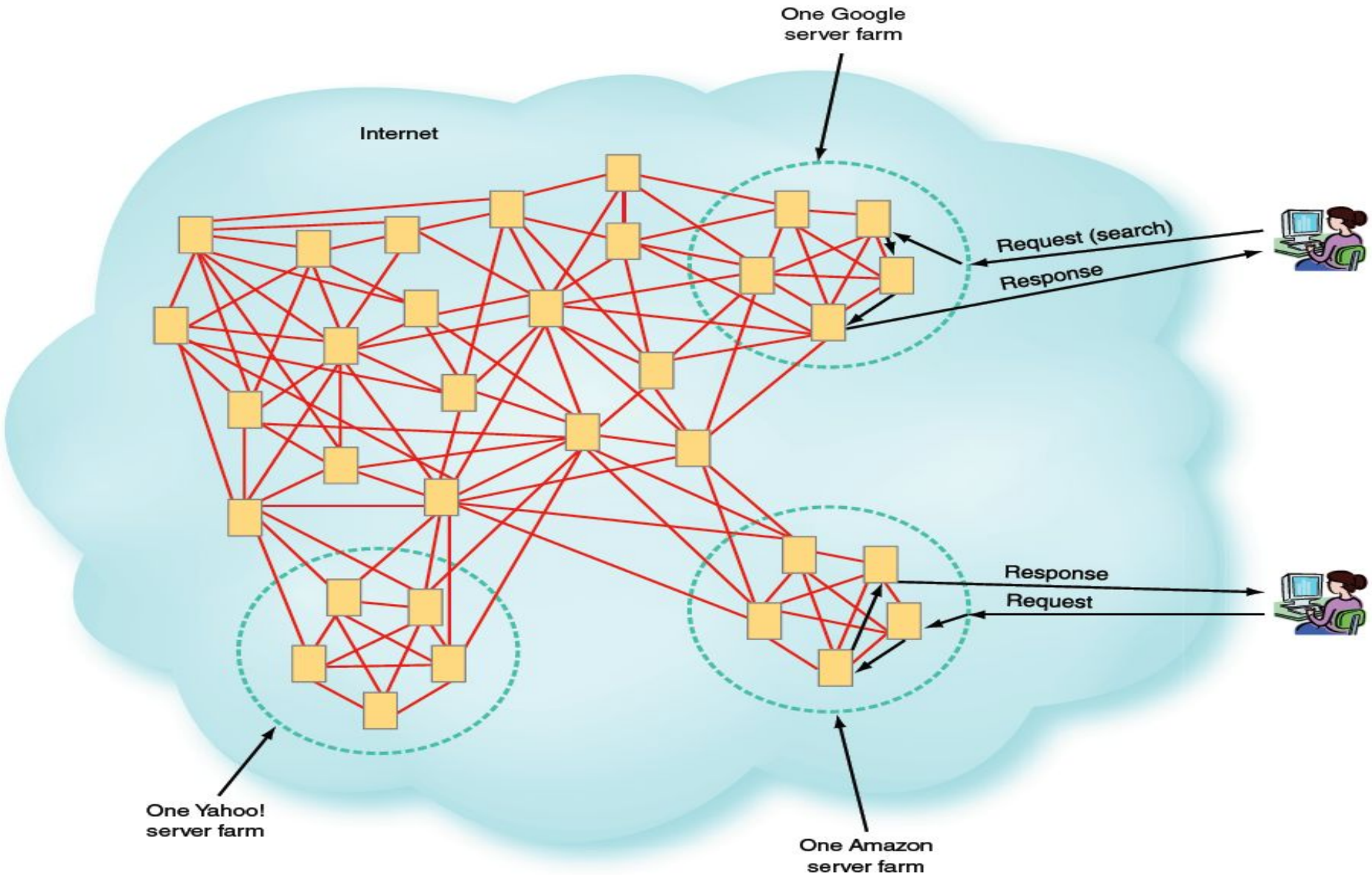


FIGURE TG 3.1 **A server farm. Notice the ventilation in the racks and ceiling.**


One Google
server farm
Internet
Request (search)
Response
Response
Request
One Yahoo!
server farm
One Amazon
server farm

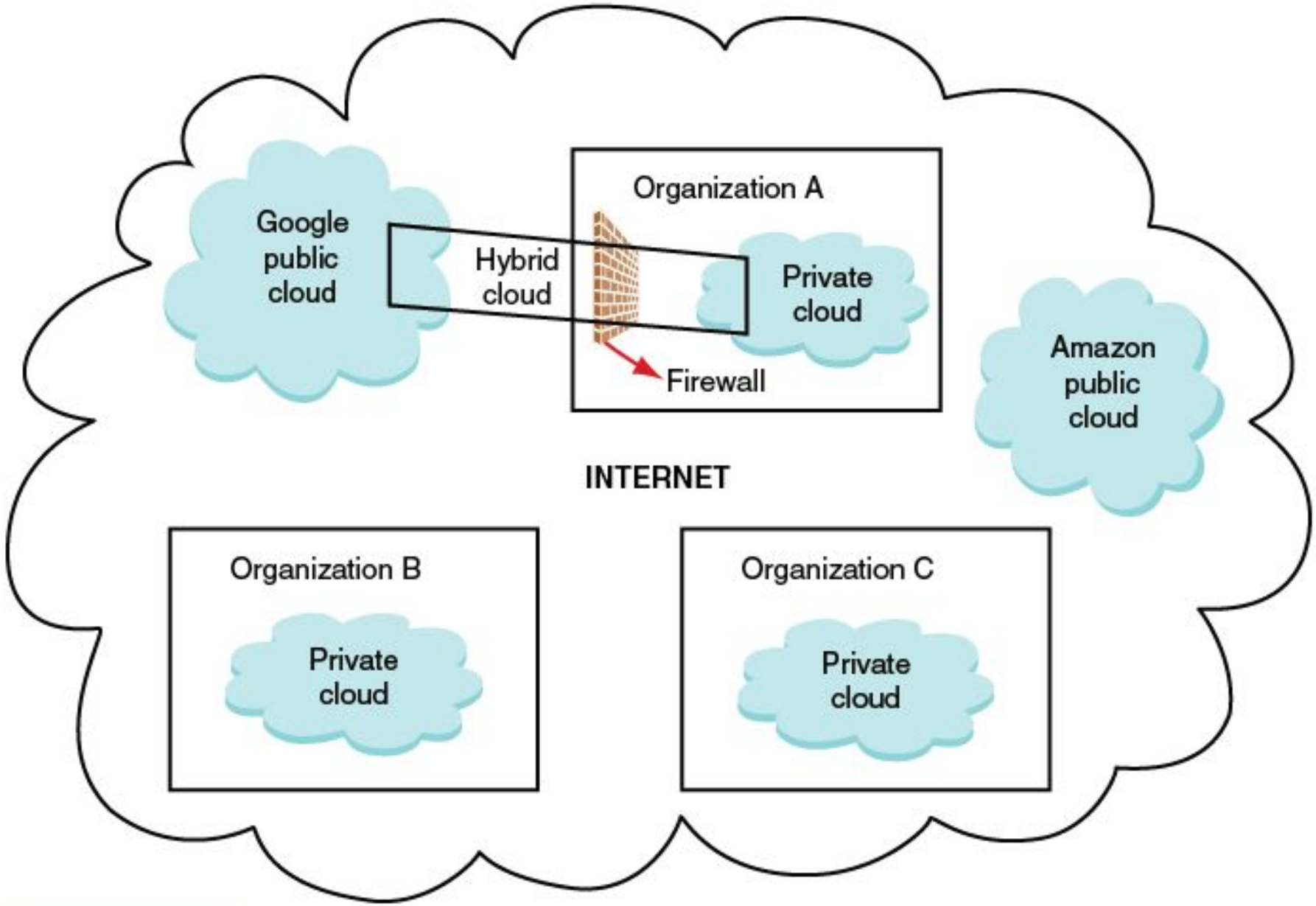


FIGURE TG 3.3 **Public clouds, private clouds, and hybrid clouds.**

Software-as-a-Service

- With the **software-as-a-service (SaaS)** delivery model, cloud computing vendors provide software that is specific to their customers' requirements. SaaS is the most widely used service model, and it provides a broad range of software applications. SaaS providers typically charge their customers a monthly or yearly subscription fee.
- In the **platform-as-a-service (PaaS)** model, customers rent servers, operating systems, storage, a database, software development technologies such as Java and .NET, and network capacity over the Internet. The PaaS model allows the customer to both run existing applications and to develop and test new applications.
- With the **infrastructure-as-a-service (IaaS)** model, cloud computing providers offer remotely accessible servers, networks, and storage capacity. They supply these resources on demand from their large resource pools, which are located in their data centres.

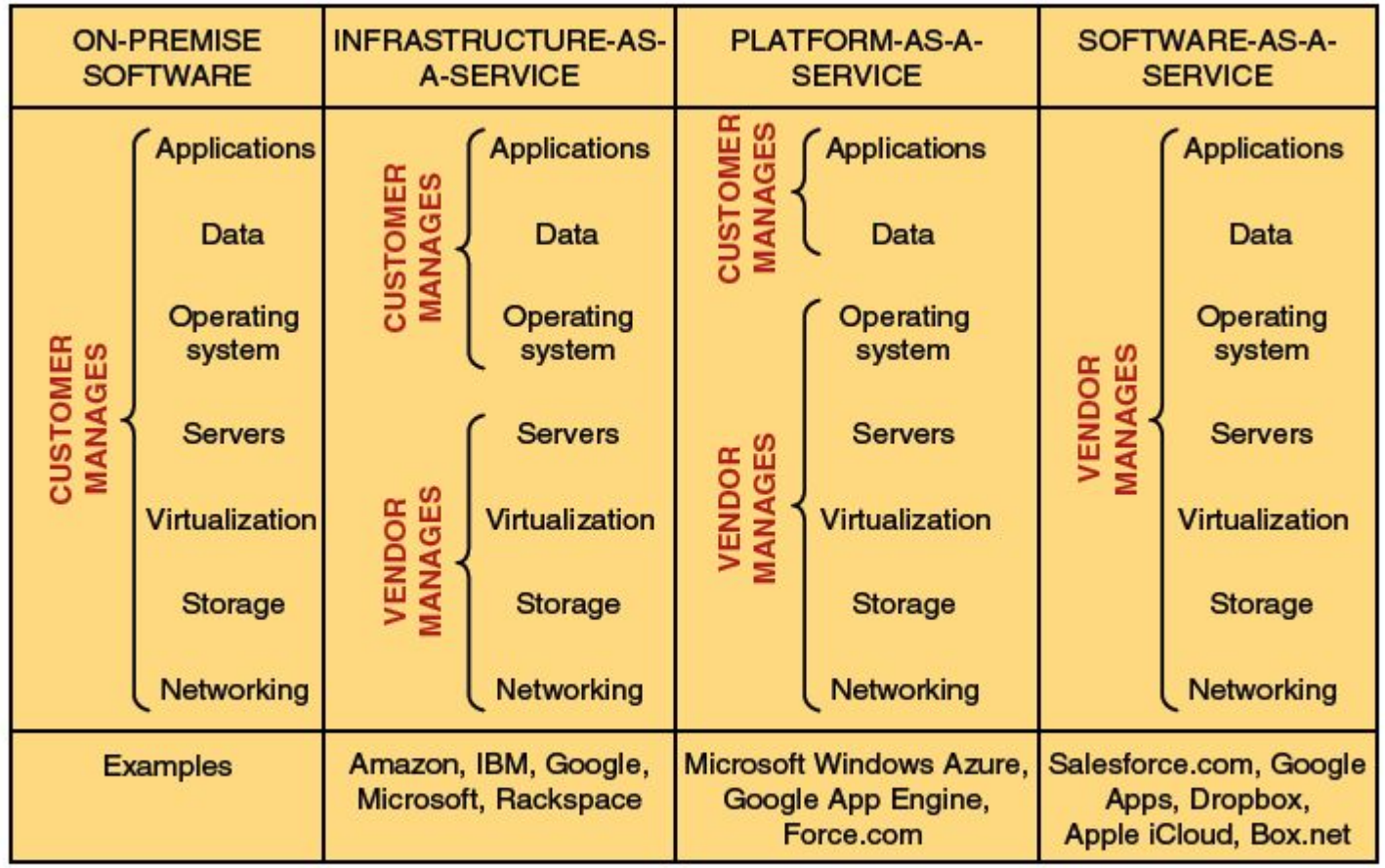
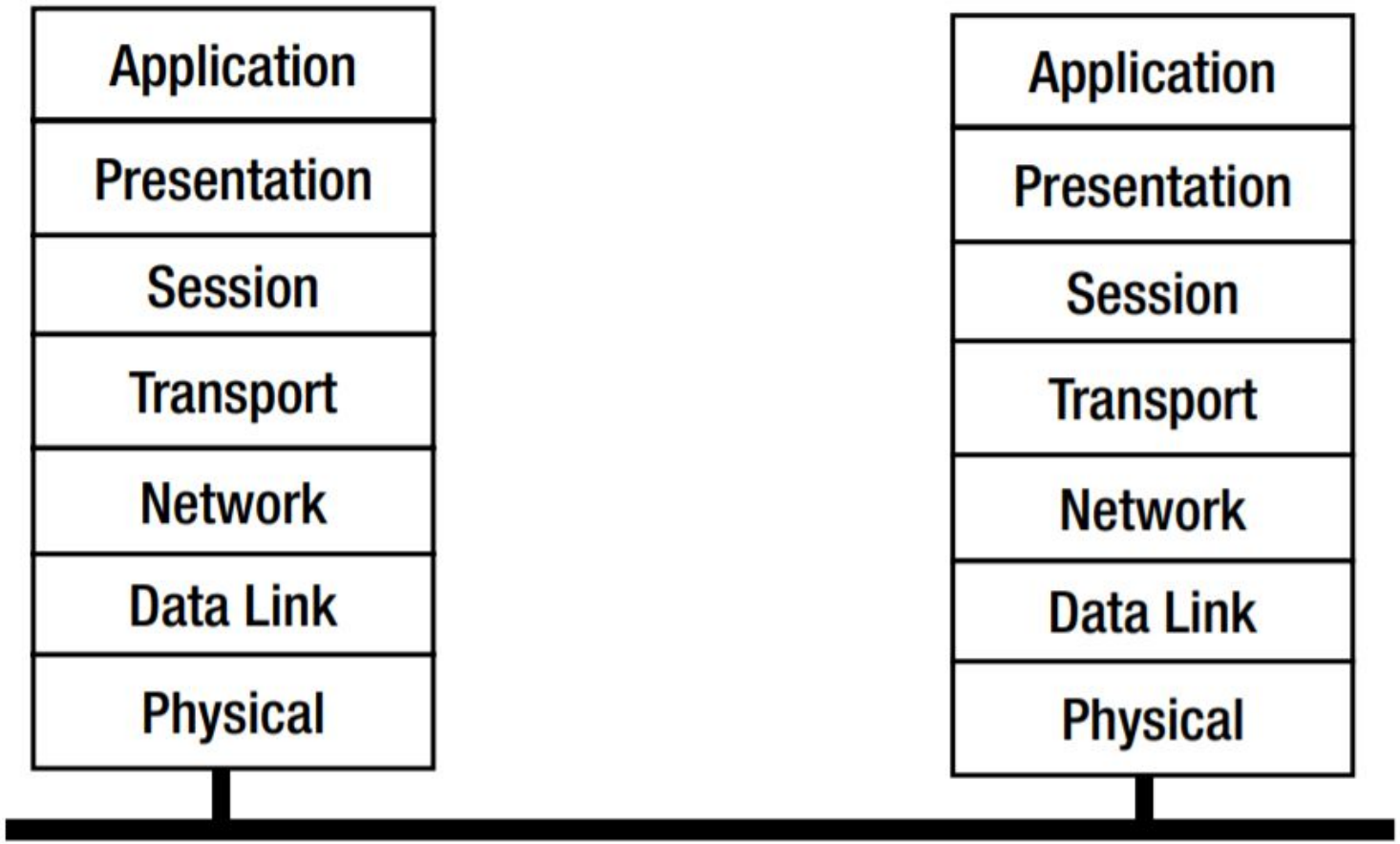


FIGURE TG 3.4 **Comparison of on-premise software, infrastructure-as-a-service, platform-as-a-service, and software-as-a-service.**

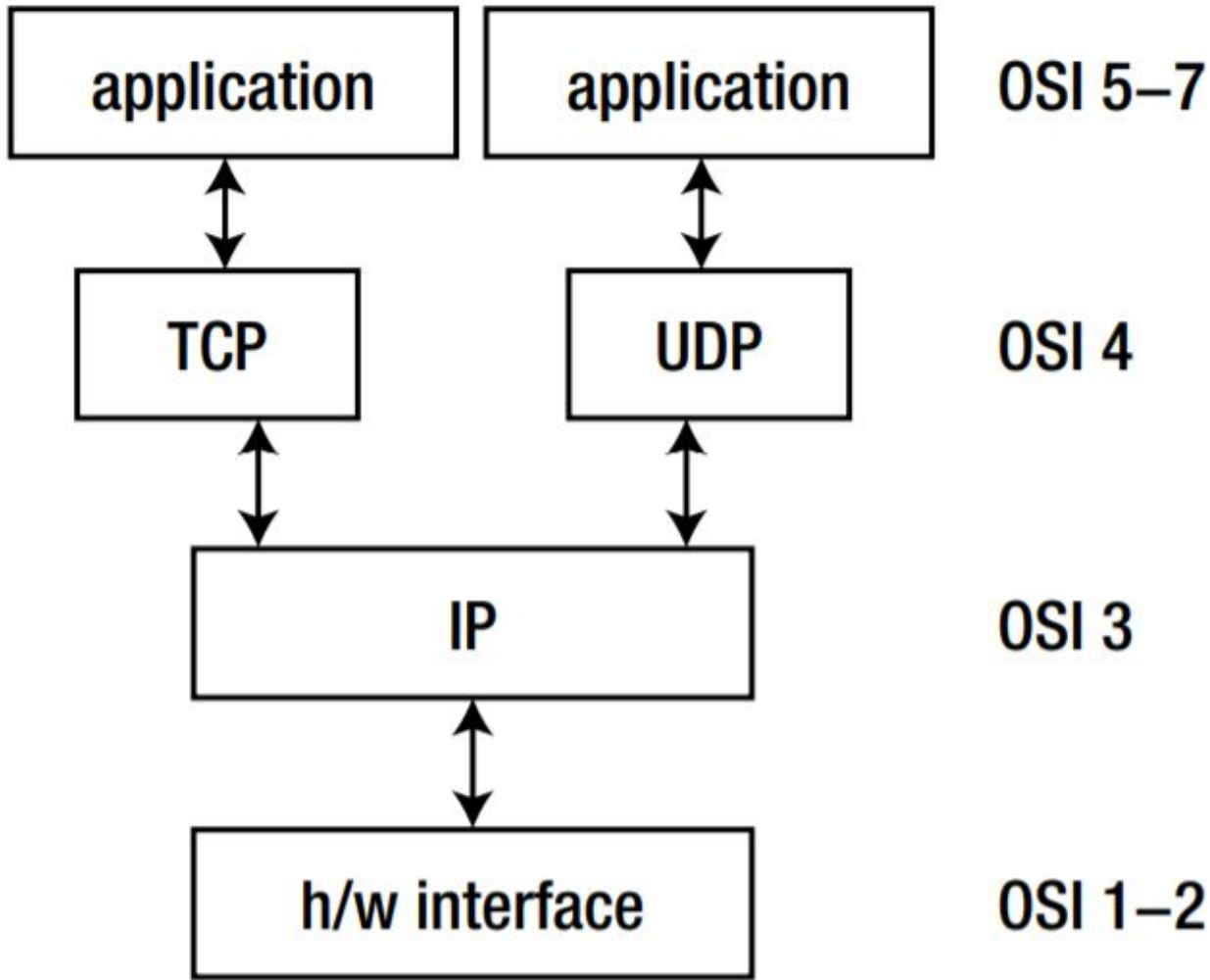
ISO OSI Protocol

Although it was never properly implemented, the OSI (Open Systems Interconnect) protocol has been a major influence in ways of talking about and influencing distributed systems design. It is commonly given as shown in Figure 1-1.



TCP/IP Protocol

While the OSI model was being argued, debated, partly implemented, and fought over, the DARPA Internet research project was busy building the TCP/IP protocols. These have been immensely successful and have led to *The Internet* (with capitals). This is a much simpler stack, as shown in Figure 1-2.



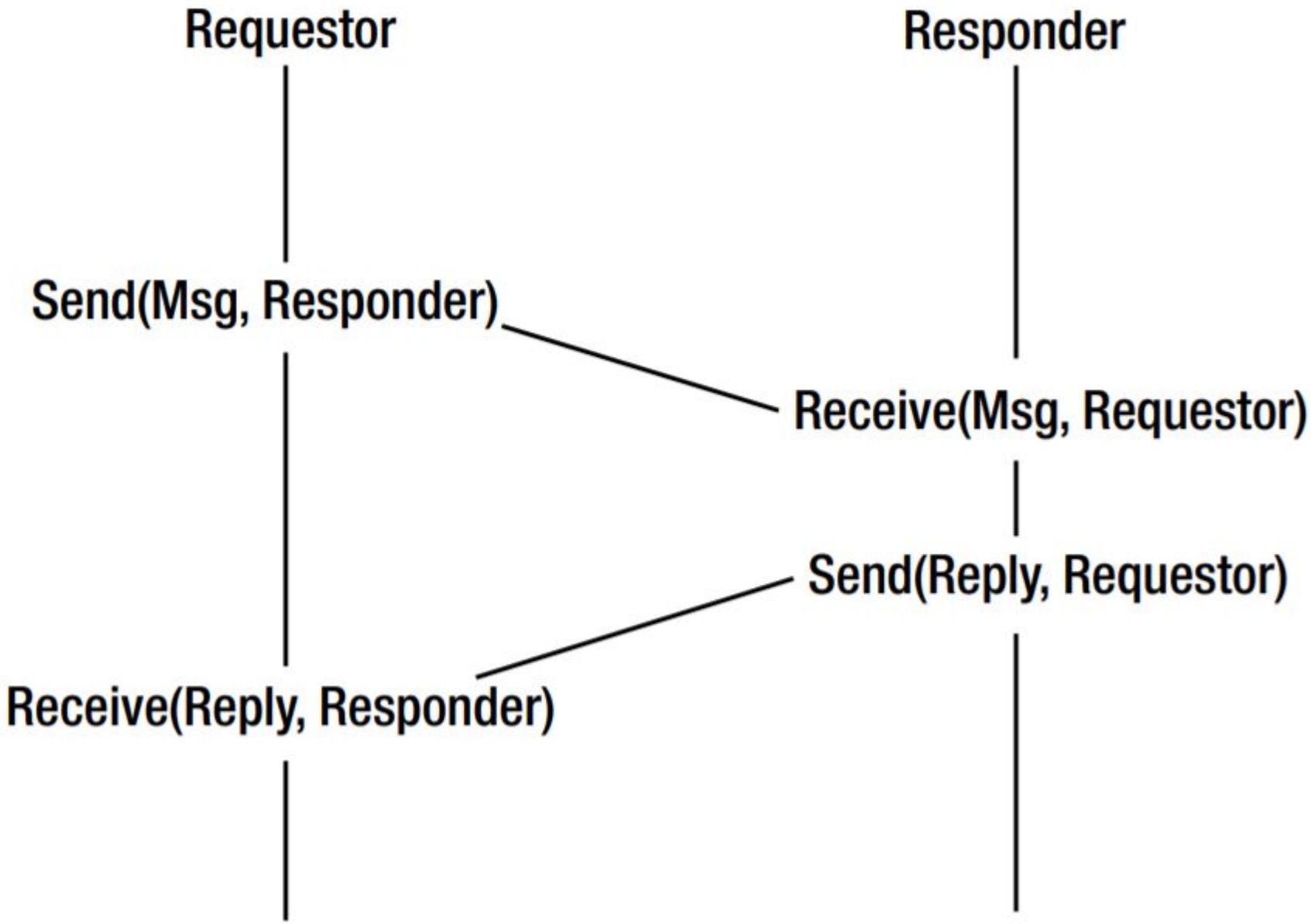


Figure 1-4. *The message passing communications model*

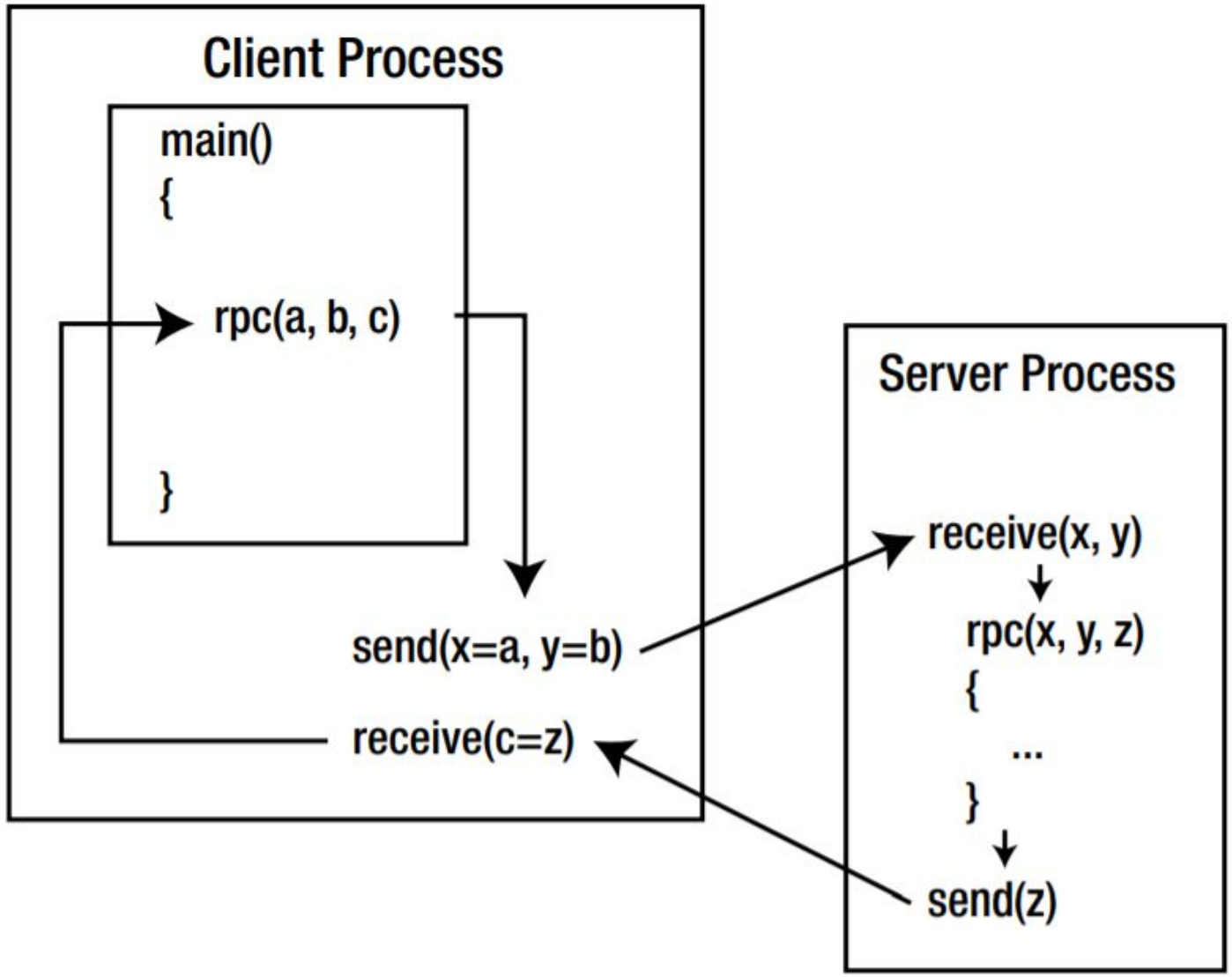
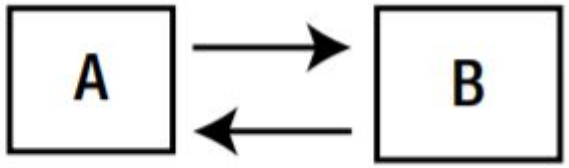


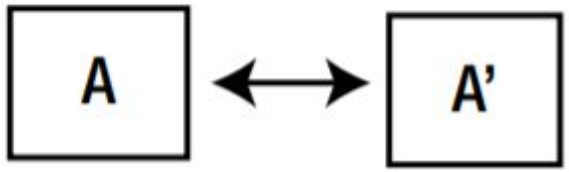
Figure 1-5. *The remote procedure call communications model*

Distributed Computing Models

client-server



peer-to-peer



If both components are equivalent, both able to initiate and to respond to messages, then we have a peer-to-peer system. Note that this is a logical classification: one peer may be a 16,000 core supercomputer, the other might be a mobile phone. But if both can act similarly, then they are peers.