

Development Strategies

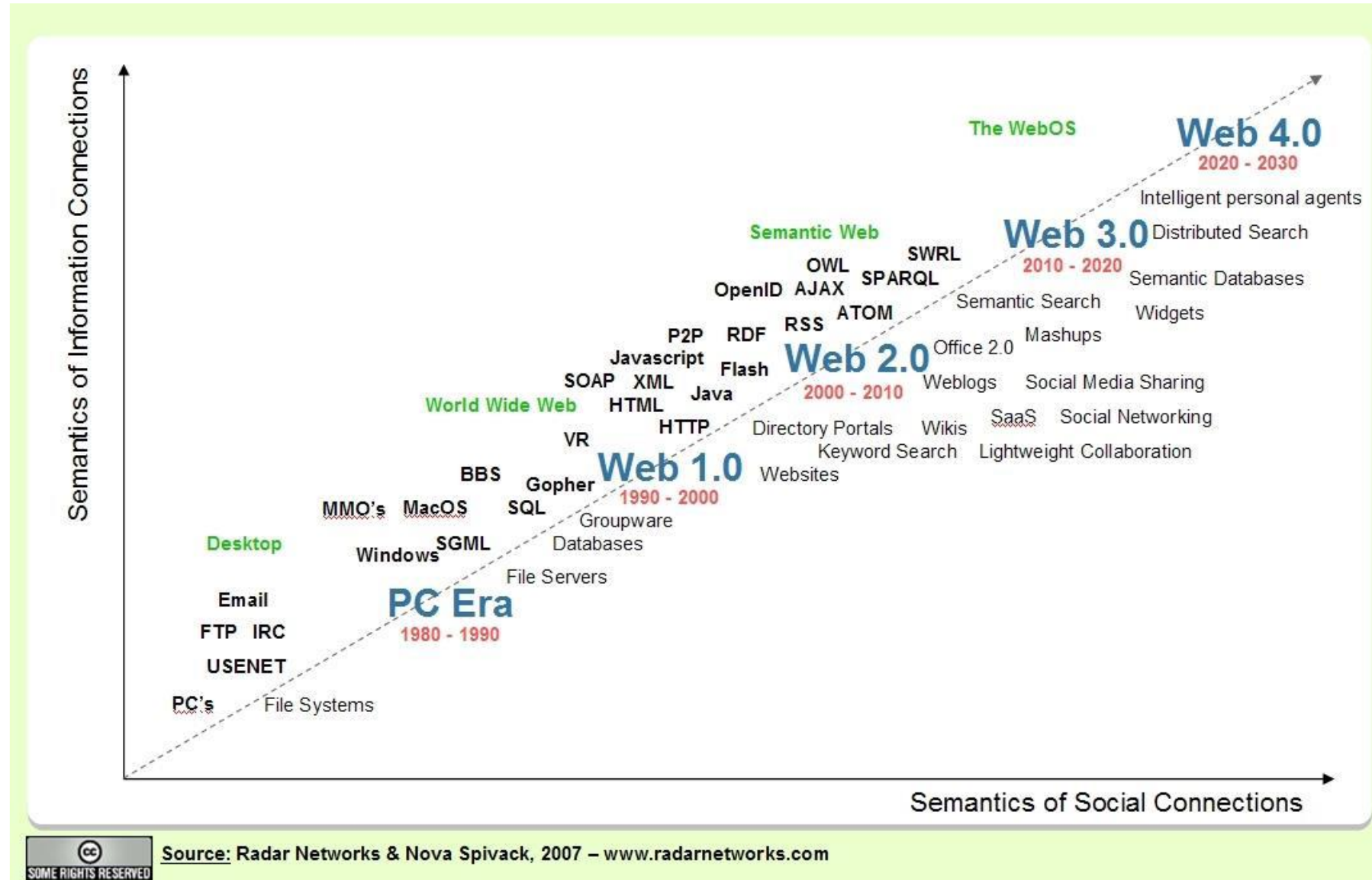
Advanced Systems Analysis & Design



At the end of the session, attendees should be able to:

- Describe the concept of Software as a Service
- Define Web 4.0 and cloud computing
- Explain software acquisition alternatives, including traditional and Web-based software development strategies
- Describe software outsourcing options, including offshore outsourcing and the role of service providers
- Explain advantages and disadvantages of in-house software development
- Discuss cost-benefit analysis and financial analysis tools
- Describe a request for proposal (RFP) and a request for quotation (RFQ)
- Describe the system requirements document
- Explain the transition from systems analysis to systems design
- Discuss systems design guidelines
- Describe software development trends

The Past, the Present, and the Future

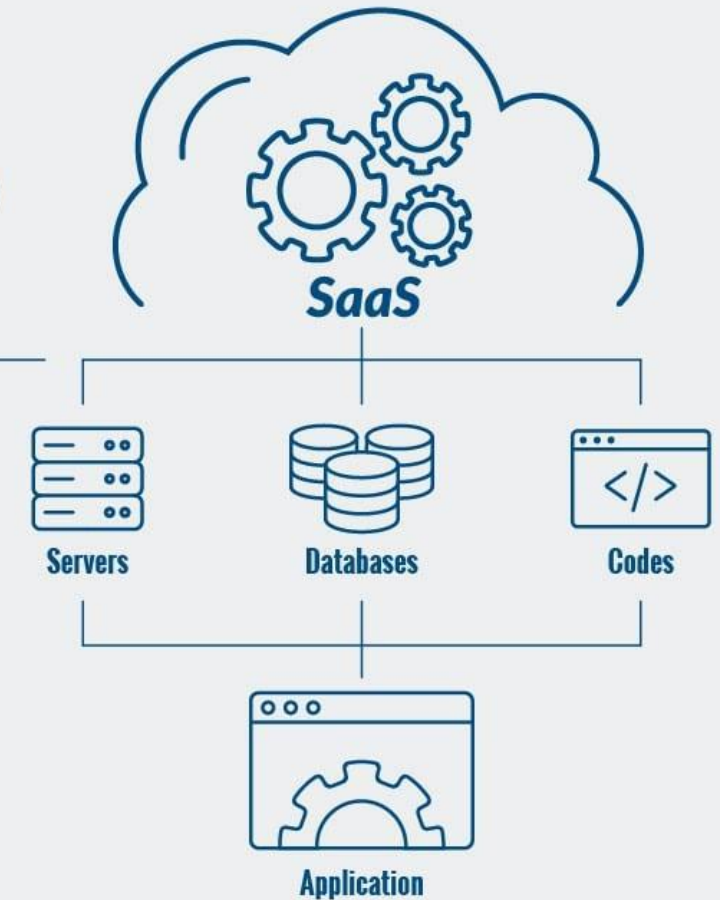


SaaS is a model of software deployment where an application is **hosted** as a service provided to customers over the Internet.

SaaS reduces the customer's need for software maintenance, operation, and support.

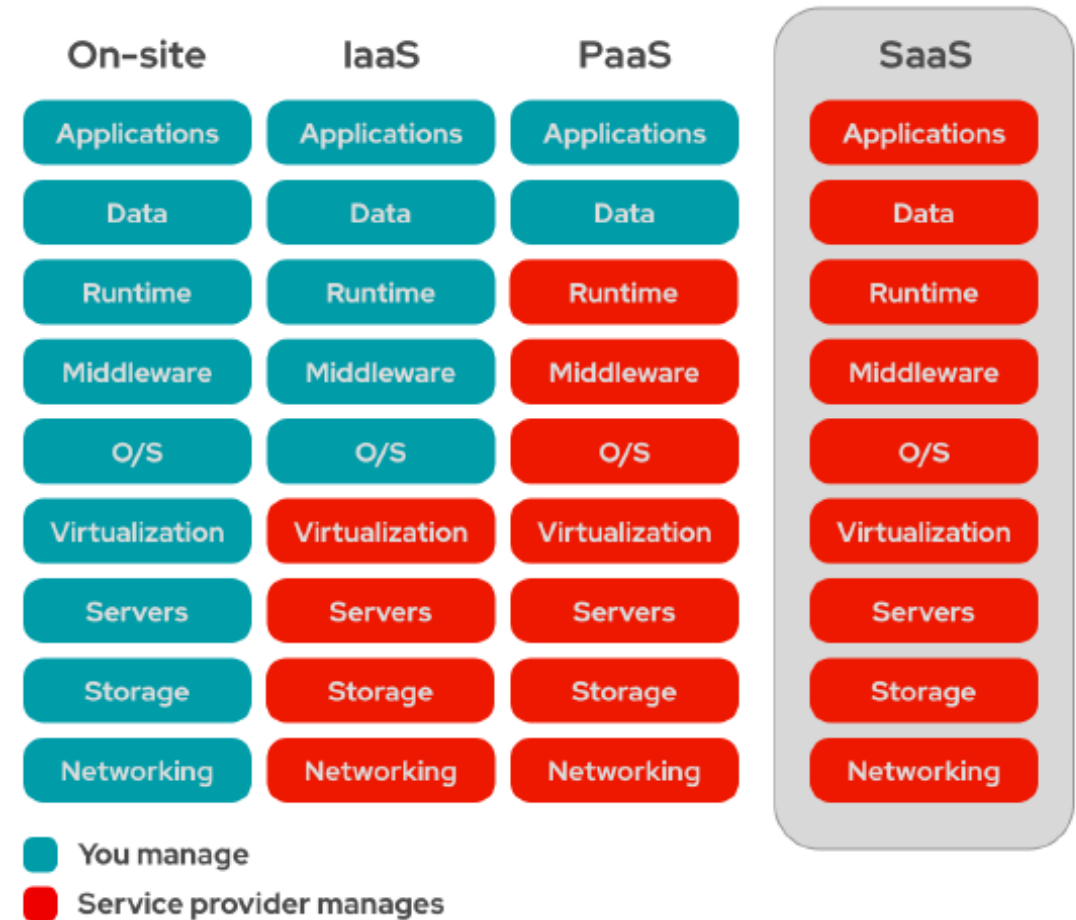
What is SaaS? ***(Infrastructure as a Service)***

Software as a Service is based entirely on the Internet, and it is an approach to software distribution by which software providers host a combination of servers, databases, and code to create applications that can be accessed by users from connected devices. Software as a Service (SaaS) brings the power of a firm's workflow to any user anywhere in the world at anytime.



Software-as-a-service (SaaS) is a form of cloud computing that delivers a cloud application—and all its underlying IT infrastructure and platforms—to end users through an internet browser.

Other approaches are Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS).



Question..

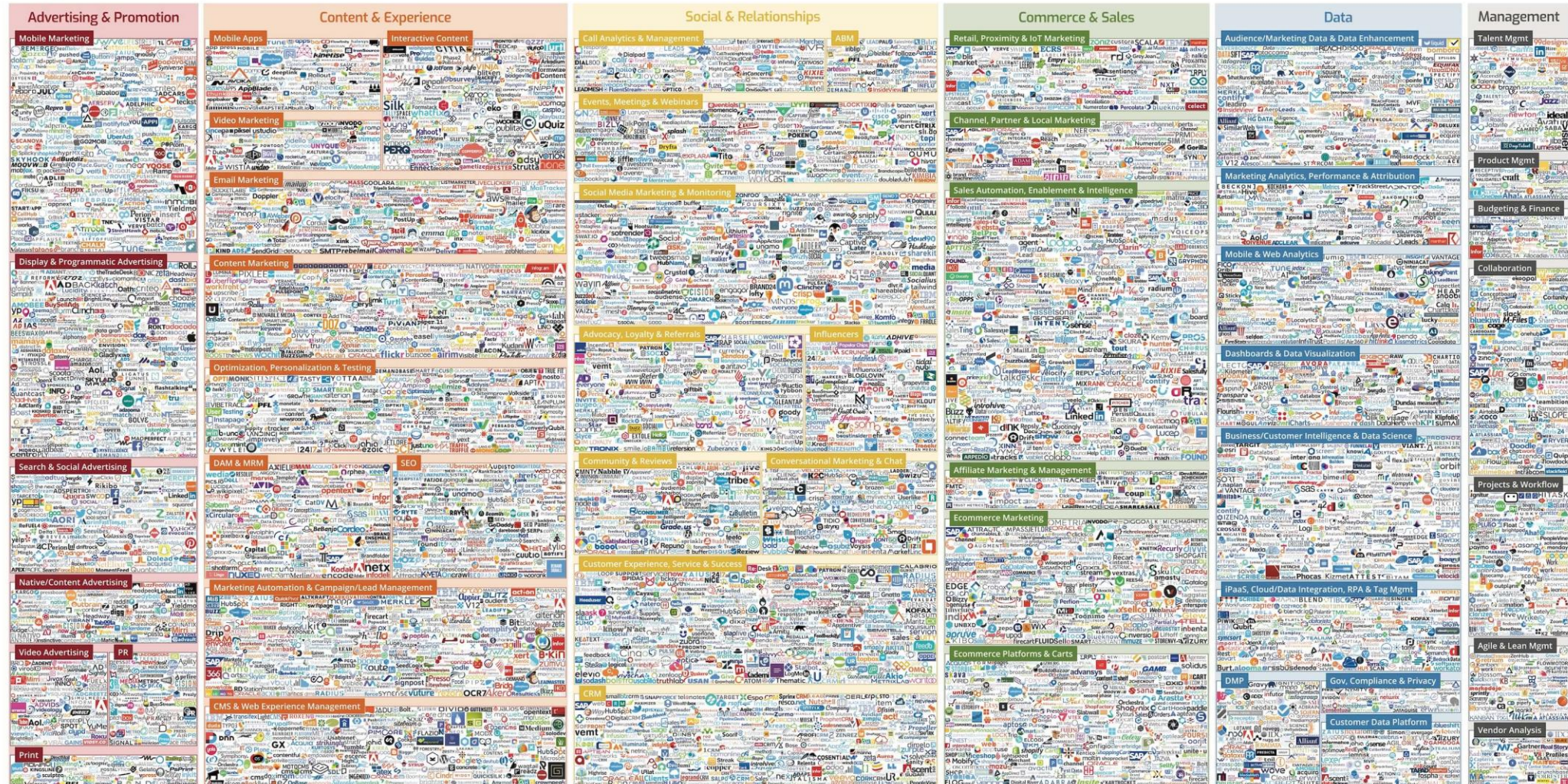
Are you using SaaS?





chiefmartec.com Marketing Technology Landscape ("Martech 5000")

April 2019



Outsourcing is the transfer of information systems development, operation, or maintenance to an outside firm that provides these services, for a fee, on a temporary or long-term basis.

Relatively minor programming tasks, the rental of software from a service provider, the outsourcing of a basic business process (often called **business process outsourcing, or BPO**), or the handling of a company's entire IT function.

Models:

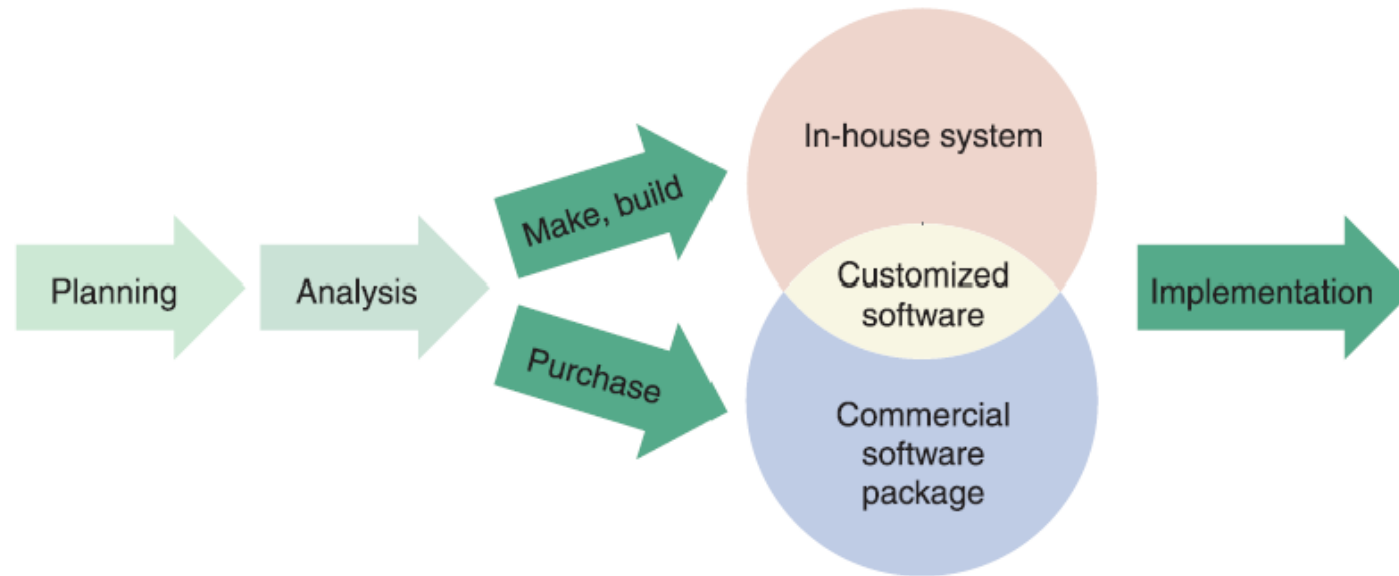
- A **fixed fee model** uses a set fee based on a specified level of service and user support.
- A **subscription model** has a variable fee based on the number of users or workstations that have access to the application.
- A **usage model** or transaction model charges a variable fee based on the volume of transactions or operations performed by the application.



A company can choose to **develop** its own systems, or **purchase**, possibly **customize**, and **implement** a software package.

A software package that can be used by many different types of organizations is called a **horizontal** application.

Software package developed to handle information requirements for a specific type of business is called a **vertical** application.



Question..

What is better in-house or purchase a software?



When to go In House?

REASONS FOR IN-HOUSE DEVELOPMENT	REASONS FOR PURCHASING A SOFTWARE PACKAGE
Satisfy unique business requirements	Lower costs
Minimize changes in business procedures and policies	Requires less time to implement
Meet constraints of existing systems	Proven reliability and performance benchmarks
Meet constraints of existing technology	Requires less technical development staff
Develop internal resources and capabilities	Future upgrades provided by the vendor
Satisfy unique security requirements	Obtain input from other companies

Three main cost analysis

- **Payback analysis:**
determines **how long** it takes an information system to pay for itself through reduced costs and increased benefits.
- **Return on investment(ROI)**
is a percentage rate that compares the total **net benefits**(the return) received from a project to the **total costs**(the investment) of the project.
- **Net present value (NPV)** of a project
is the **total value** of the **benefits minus** the **total value** of the **costs**, with both costs and benefits adjusted to reflect the point in time at which they occur.



Step 1: Evaluate the Information System Requirements

- Identify key features
- Consider network and web-related issues
- Estimate volume and future growth
- Specify hardware, software, or personnel constraints
- Prepare a request for proposal or quotation
 - A **request for proposal (RFP)** is a document that describes your company, lists the IT **services** or **products** you **need**, and specifies the **features** you **require**.
 - A **request for quotation (RFQ)** is more specific than an RFP. When you use an RFQ, you already know the specific product or service you want, and you need to obtain price quotations or bids.
 - An **evaluation model** is a technique that uses a common yardstick to measure and compare vendor ratings.

Weighted Evaluation Model for a Network Project

Instructions: Rate each vendor on a scale from 1 (low) to 10 (high), then multiply the vendor's score by the weight factor. Add vendor scores to calculate total points.

	WEIGHT FACTOR	VENDOR A	VENDOR B	VENDOR C
Price	25	$6 * 25 = 150$	$5 * 25 = 125$	$9 * 25 = 225$
Completion Date	25	$2 * 25 = 50$	$5 * 25 = 125$	$8 * 25 = 200$
Layout/Design	35	$8 * 35 = 280$	$8 * 35 = 280$	$5 * 35 = 175$
References	15	$10 * 15 = 150$	$6 * 15 = 90$	$3 * 15 = 45$
TOTAL POINTS	100	630	620	645

Step 2: Identify Potential Vendors or Outsourcing Options

Step 3: Evaluate the Alternatives

- Existing users
- Application testing
- Benchmarking
 - A benchmark measures the time a package takes to process a certain number of transactions. For example, a benchmark test can measure the time needed to post 1,000 sales transactions.

Step 4: Perform Cost-Benefit Analysis

Step 5: Prepare a Recommendation

Step 6: Implement the Solution

- The **system requirements document**, or **software requirements specification**, contains the requirements for the new system, describes the alternatives that were considered, and makes a specific recommendation to management.
- This important document is the starting point for measuring the performance, accuracy, and completeness of the finished system before entering the systems design phase.
- The system requirements document is like a **contract** that identifies what the system developers must deliver to users.

Software Requirement Specifications



Logical and Physical Design

A **logical** design defines what must take place, not how it will be accomplished.
Does not address the actual methods of implementation.

A **physical** design is like a set of blueprints for the actual construction of a building.
Describes the actual processes of entering, verifying, and storing data;
The physical layout of data files and sorting procedures, the format of reports, and so on.

The goal of systems design is to build a system that satisfies business requirements.

A successful system must be effective, reliable, and maintainable:

- A system is **effective** if it supports business requirements and meets user needs.
- A system is **reliable** if it handles input errors, processing errors, hardware failures, or human mistakes. A good design will anticipate errors, detect them as early as possible, make it easy to correct them, and prevent them from damaging the system itself.
- A system is **maintainable** if it is flexible, scalable, and easily modified. Changes might be needed to correct problems, adapt to user requirements, or take advantage of new technology.

Prototyping produces an early, rapidly constructed working version of the proposed information system, called a prototype.

Prototyping, which involves a repetitive sequence of analysis, design, modeling, and testing, is a common technique that can be used to design anything from a new home to a computer network.

User input and feedback is essential at every stage of the systems development process.

Prototyping allows users to examine a model that accurately represents system outputs, inputs, interfaces, and processes.

Users can “test-drive” the model in a risk-free environment and either approve it or request changes. In some situations, the prototype evolves into the final version of the information system; in other cases, the prototype is intended only to validate user requirements and is discarded afterward.

The most intense form of prototyping occurs when **agile** methods are used.

- As the agile process continues, developers revise, extend, and merge earlier versions into the final product.
- An agile approach emphasizes continuous feedback, and each incremental step is affected by what was learned in the prior steps.

Rapid throwaway

This method involves **exploring ideas** by **quickly developing** a **prototype** based on preliminary requirements that is then revised through customer feedback.

The name rapid throwaway refers to the fact that each prototype is completely **discarded** and may **not** be a **part** of the **final product**.

Evolutionary

This approach uses a **continuous, working prototype** that is **refined** after each iteration of customer feedback. Uses RAD (Rapid Application Development) methodology.

Because each prototype is not started from scratch, this method saves time and effort.

Incremental

This technique **breaks** the concept for the **final product** into **smaller pieces**, and prototypes are created for each one. In the end, these prototypes are **merged** into the final product.

Extreme

This prototype model is used specifically for web development.

All web prototypes are built in an HTML format with a services layer and are then integrated into the final product.

Advantages:

- Users and systems developers can **avoid misunderstandings**.
- System developers can create **accurate specifications** for the finished system based on the prototype.
- Managers can **evaluate** a **working model** more effectively than a paper specification.
- Systems analysts can use a prototype to develop **testing** and **training procedures** before the finished system is available.
- Prototyping **reduces** the **risk** and potential **financial exposure** finished system fails to support business needs.

Disadvantages:

- The rapid pace of development can create **quality problems**, which are not discovered until the finished system is operational.
- Other system requirements, such as **reliability** and **maintainability**, **cannot** be **tested** adequately using a prototype.
- In very **complex systems**, the prototype becomes unwieldy and **difficult** to **manage**.
- Confusing the **prototype** with the **final product**.


Bottom-Up Model (part to whole)

- A system design approach where **parts** of the system are defined in detail.
- Once these parts are designed and developed, then these parts or components are **linked together** to prepare a bigger component.
- This approach is **repeated** until the complete system is built.
- Advantage of Bottom-Up Model is in making decisions at very low level and to decide **the re-usability** of components.
- Example: **Evolutionary Prototyping Model**

Top-Down Model (whole to parts)

- A system design approach where design **starts from the system as a whole**.
- Complete system is then **divided** into smaller sub-applications with more details.
- Each part again goes through the top-down approach till the complete system is designed with all minute details.
- It is also termed as breaking the bigger problem into smaller problems and solving them individually in recursive manner.
- Example: **Incremental Prototyping Model**

Object Oriented Programming

OOP	VS	POP
		
<ul style="list-style-type: none">• Object-oriented.• The program is divided into objects.• Bottom-up approach.• Inheritance property is used.• It uses an access specifier.• Encapsulation is used to hide the data.• Concept of virtual function.• C++, Python, Java.		<ul style="list-style-type: none">• Structure oriented.• Program is divided into functions.• Top-down approach.• Inheritance is not allowed.• It doesn't use an access specifier.• No data hiding is done.• No virtual function.• C, Pascal.

Procedural Oriented Programming



Thank you!