Training Deck for Implementation of ERP and EHR Systems

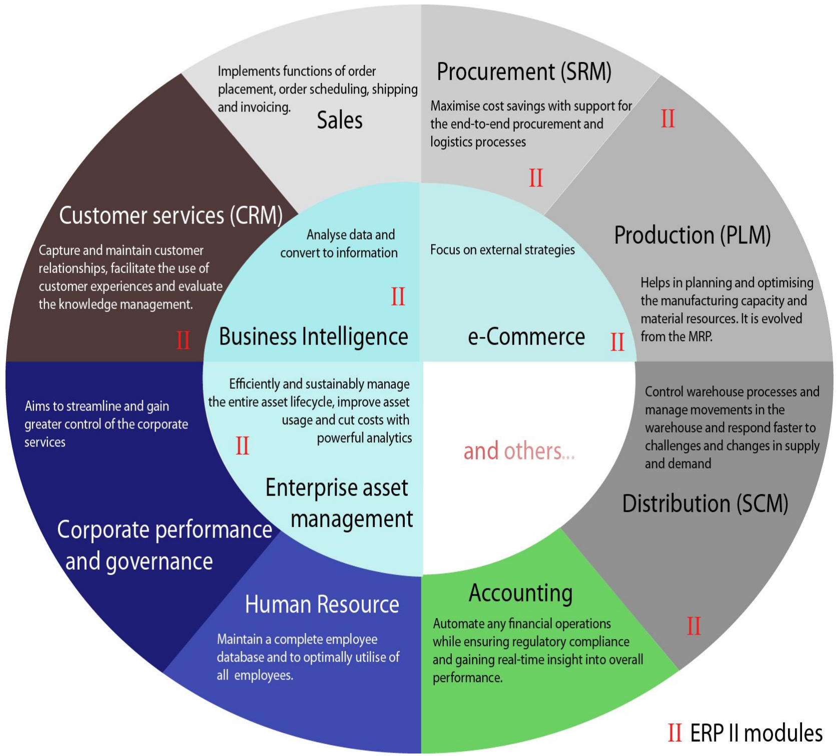
Yiqiao Yin

Architecture of Enterprise Resource Solutions (ERP)

What exactly is Enterprise Resource Planning or ERP?

Enterprise Resource Planning or ERP is the centralized management system integrated every important part of the business. It can include a wide range of business components inside a company such as human resource, accounting, production, sales, technology, research & development, and so on. The most important takeaways for ERP can be summarized below:

- ERP solutions are now web-based and the common ERP software can be integrated throughout all components of a company.
- The biggest benefit of ERP is the free flow of communication across all business functions and the freedom of collaboration cross different functions of a company.
- A company's management team is likely facing issues if there is not a complete ERP system or if the existing ERP system is failing.



Implements functions of order placement, order scheduling, shipping and invoicing.
Sales
Procurement (SRM)
Maximise cost savings with support for the end-to-end procurement and logistics processes
II
II
Customer services (CRM)
Capture and maintain customer relationships, facilitate the use of customer experiences and evaluate the knowledge management.
II
Analyse data and convert to information
II
Business Intelligence
Focus on external strategies
e-Commerce
II
Production (PLM)
Helps in planning and optimising the manufacturing capacity and material resources. It is evolved from the MRP.
Aims to streamline and gain greater control of the corporate services
Efficiently and sustainably manage the entire asset lifecycle, improve asset usage and cut costs with powerful analytics
II
Enterprise asset management
and others...
Control warehouse processes and manage movements in the warehouse and respond faster to challenges and changes in supply and demand
Distribution (SCM)
Corporate performance and governance
Human Resource
Maintain a complete employee database and to optimally utilise of all employees.
Accounting
Automate any financial operations while ensuring regulatory compliance and gaining real-time insight into overall performance.
II
II ERP II modules

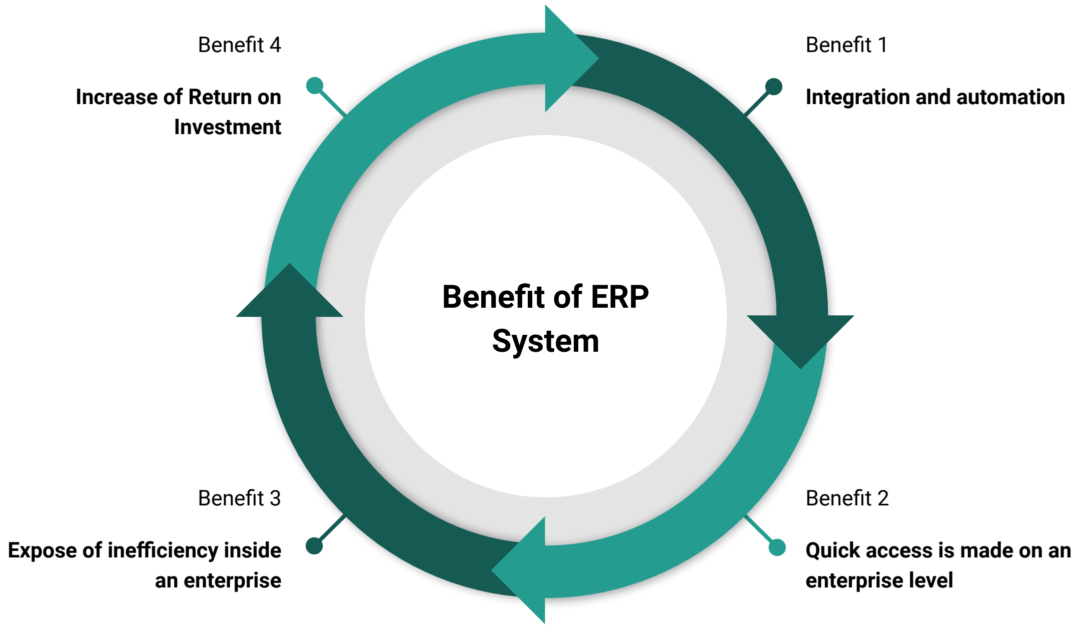
The link of the figure is here.

Yiqiao Yin | YouTube

Architecture of Enterprise Resource Solutions (ERP)

Major benefits of ERP includes the following:

- Integration and automation of the ERP system at an enterprise level allows business expansion, cost reduction, and improvement of business operation on a management level.
- Quick access is made available and possible for each sub-division of an enterprise given successfully built ERP system.
- Expose of inefficiency of a poorly managed division inside a company.
- Proven records of increasing Return on Investment (ROI) as well as increasing productivity with reduced administrative costs.


Benefit 1
Integration and automation
Benefit 2
Quick access is made on an enterprise level
Benefit 3
Expose of inefficiency inside an enterprise
Benefit 4
Increase of Return on Investment
Benefit of ERP System

Architecture of Electronic Health Patient Records (EHR)

An Electronic Health Record or EHR is an internet-based patient health record and medical record. This is a system that manages data workflow for patients and providers over time. The database should include all of the important administrative clinical data that helps tracks the patients information including demographics, progress, problems, medications, and so on.

The link of the figure is here.

Architecture of Electronic Health Patient Records (EHR)

Reduce System Errors

A successful EHR system helps minimize the errors created during elongated process of the modern day healthcare system. The EHR helps improve the accuracy and clarity of the patient's' medical records.

Availability of Health Information

A powerful EHR system makes the patients health information available. This is to allow duplication of tests and unnecessary payments. This could even prevent delay of medications and potentially help serve patients and doctors to make better decisions.

Reduce Medical Errors

Just like reduction of system errors, a successful EHR system also reduces medical errors. This can mean enormous amount of medical costs imposed on the patients from the insurance company.

Description of the Architecture of Object-oriented Programming Languages

A little history ...

Object-oriented programming or OOP has becoming an important tool to master in large-scale software system development. The OOP is usually constructed using tight arguments and flexibly defined variables to orchestrate the production pipeline.

The nature of OOP regards to the meaning of a statement. Consider a simple statement in human language. It is a functional form that stores stories. These stories are information that can be used to construct maps.

Description of the Architecture of Object-oriented Programming Languages

To understand the architecture of OOP, the first thing to mention is ALGOL-like languages. We often times proceed with 3 steps.

First, we separate stores within blocks in order to declare the necessary components inside an object. This way it is possible to create multiple instances and these instances form blocks of coding structure to be used later on to construct a model. A model, often times in functional form, is then executed as an algorithm.

Second, there is intrinsic connection between blocks. Essentially, communicational tools need to be built of which another name you might recall is “shared variables”. “Shared variables” allow programs to talk to each other, which mathematically can be understood as composite functions.

Finally, complicated OOP usually involves multiple stages of the first and the second components described above. In other words, a there is a need to call by reference and the feature names need to be well defined across different programs. When the programs are compiled, the execution needs to be mapped accordingly and correctly. Python is a famous type of OOP and there are many famous modules in Python that essentially incorporates the first and the second steps discussed above. A famous deep learning library called Keras can serve as an example here.

Description of the Architecture of Object-oriented Programming Languages

01 **Abstraction**

- Environment is not necessary and no abstraction mechanism
- Data abstraction needs to be defined separately
- Abstraction and encapsulation both required for data control

02 **Inheritance**

- Classes describe the similarities amongst different objects and many of which can be associated with inheritance
- Single inheritance system exists amongst superclasses
- Does not require any changes in semantical area

03 **Polymorphism**

- A situation can occur in different forms where the interface may remain the same
- Potential “instanceof” test can be developed to test if an object is polymorphic
- Static vs. Dynamic

04 **Encapsulation**

- Abstraction and encapsulation both required for data control
- Designed to hide inherited instance variables
- Objects with invoked methods can lead to unwanted breaches of encapsulation
- Encapsulation is an important principle when designing internal construction of an object or class

05 **Reusable Components**

- Often times used with encapsulation because information hidden can be retracted
- Requires on-line searching mechanism to a large extent

Data Management across Modules

One important field of application is the gigabases of DNA and RNA sequencing data the current technology is able to generate in a day or a week. These types of business modules all require successful database management systems or also known as DBMS. The success of tomorrow's life science based technological advancement all depend on today's capability to provide successful DBMS systems, which essentially focuses on efficient management pipeline across different modules (Schadt, 2010).

There are two aspects to mention: cloud-based computing and heterogeneous computing. Both are essentially to tackle today's big data problems at an enterprise level.

Cloud-based computing:

Cloud-based computing system has been the major force of the evolution in high-speed, low-cost computing environment. However, based on empirical practice, it is not entirely feasible to deploy all the data to the cloud due to limited service provider and geopolitical complications. The cloud computing technology equips users the capability to construct abstract layers of software programmers and deploy it remotely. The systems enable convenience and on-demand application requirements. However, the system is not perfect yet. There are constant on-going issues with data security and the bandwidth of large-scale datasets is not usually accessible at very rural area.

Heterogeneous computing:

The computers can be integrated with different accelerators such as GPU or TPU. The former is called Graphical Processing Unit while the latter is Tensor Processing Unit. Both are widely used in modern day scientific computing. There are many significant expertise programmers required to utilize and design software products that satisfy the needs of ongoing demand for this part of the service. The power of cloud-based technology enables cluster service and parallel training when it comes to handling complex modeling architecture in learning about the complicated structure of DNA chain.

Data Management across Modules

The hardware and software stacks together to compose a complicated multi-layer system (Schadt, 2010).

There are two aspects to mention: cloud-based computing and heterogeneous computing. Both are essentially to tackle today's big data problems at an enterprise level.

Cloud-based computing:

Cloud-based computing system has been the major force of the evolution in high-speed, low-cost computing environment. However, based on empirical practice, it is not entirely feasible to deploy all the data to the cloud due to limited service provider and geopolitical complications. The cloud computing technology equips users the capability to construct abstract layers of software programmers and deploy it remotely. The systems enable convenience and on-demand application requirements. However, the system is not perfect yet. There are constant on-going issues with data security and the bandwidth of large-scale datasets is not usually accessible at very rural area.

Heterogeneous computing:

The computers can be integrated with different accelerators such as GPU or TPU. The former is called Graphical Processing Unit while the latter is Tensor Processing Unit. Both are widely used in modern day scientific computing. There are many significant expertise programmers required to utilize and design software products that satisfy the needs of ongoing demand for this part of the service. The power of cloud-based technology enables cluster service and parallel training when it comes to handling complex modeling architecture in learning about the complicated structure of DNA chain.

Famous Examples

Procedural Programming is derived from structured programming. The program is built using parts with routines and subroutines. Object-oriented programming is built upon objects. The objects have attributes and the objects act like building blocks. We present the tabular form of comparison below.

	Procedural Programming	**Object-Oriented Programming**
Parts	Parts act like building block.	Objects act like building block.
Direction	Top-down	Bottom-up
Access specifier	No	Yes
Adding new attributes	Hard	Easy
Security	Less secure	More secure

Famous Examples

Procedural Programming is derived from structured programming. The program is built using parts with routines and subroutines. Object-oriented programming is built upon objects. The objects have attributes and the objects act like building blocks. We present the technical form of comparison below.

```
# Program to display Fibonacci sequence in FORTRAN

PROGRAM    Fibonacci
  IMPLICIT   NONE
  INTEGER :: FIRST_NUM, SECOND_NUM, TEMP, IX
  FIRST_NUM = 0
  SECOND_NUM = 1
  WRITE (*,*) FIRST_NUM
  WRITE (*,*) SECOND_NUM
  DO IX = 1, 45, 1
    TEMP = FIRST_NUM + SECOND_NUM
    FIRST_NUM = SECOND_NUM
    SECOND_NUM = TEMP
    WRITE (*,*) TEMP
  END DO
END PROGRAM Fibonacci
```

```
# Program to display the Fibonacci sequence in Python

nterms = int(input("How many terms? "))

# first two terms
FIRST_NUM, SECOND_NUM = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return FIRST_NUM
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(FIRST_NUM)
# generate fibonacci sequence
else:
  print("Fibonacci sequence:")
  while count < nterms:
     print(FIRST_NUM)
     nth = FIRST_NUM + SECOND_NUM
     # update values
     FIRST_NUM = SECOND_NUM
     SECOND_NUM = nth
     count += 1
```

Potential Recommendations

References

Bruce, K. B. (1994). A paradigmatic object-oriented programming language: Design, static typing and semantics. *Journal of Functional Programming*, *4*(2), 127-206.

El-Atawy, S. S., & Khalefa, M. E. (2016, May). Building an ontology-based electronic health record system. In *Proceedings of the 2nd Africa and Middle East Conference on Software Engineering* (pp. 40-45).

Grossniklaus, M., Leone, S., Spindler, A. D., & Norrie, M. C. (2010, June). Dynamic metamodel extension modules to support adaptive data management. In *International Conference on Advanced Information Systems Engineering* (pp. 363-377). Springer, Berlin, Heidelberg.

Kilov, H. (1990, April). From semantic to object-oriented data modeling. In *Systems Integration'90. Proceedings of the First International Conference on Systems Integration* (pp. 385-393). IEEE.

MacKinnon, W., & Wasserman, M. (2009, January). Integrated electronic medical record systems: Critical success factors for implementation. In *2009 42nd Hawaii International Conference on System Sciences* (pp. 1-10). IEEE.

Schadt, E. E., Linderman, M. D., Sorenson, J., Lee, L., & Nolan, G. P. (2010). Computational solutions to large-scale data management and analysis. *Nature reviews genetics*, *11*(9), 647-657.

Wolczko, M. I., & Jones, C. (1988). *Semantics of object-oriented languages* (Doctoral dissertation, University of Manchester).