**SYSTEMS MODELING**

**Systems modeling** or **system modeling** is the [interdisciplinary study](https://en.wikipedia.org/wiki/Interdisciplinarity) of the use of [models](https://en.wikipedia.org/wiki/Scientific_modeling) to conceptualize and construct [systems](https://en.wikipedia.org/wiki/System) in [business](https://en.wikipedia.org/wiki/Business) and [IT development](https://en.wikipedia.org/wiki/Information_technology).

A common type of systems modeling is [function modeling](https://en.wikipedia.org/wiki/Function_model), with specific techniques such as the [Functional Flow Block Diagram](https://en.wikipedia.org/wiki/Functional_Flow_Block_Diagram) and [IDEF0](https://en.wikipedia.org/wiki/IDEF0). These models can be extended using [functional decomposition](https://en.wikipedia.org/wiki/Functional_decomposition), and can be linked to [requirements](https://en.wikipedia.org/wiki/Requirement) models for further systems partition.

Contrasting the functional modeling, another type of systems modeling is [architectural modeling](https://en.wikipedia.org/wiki/Systems_architecture) which uses the [systems architecture](https://en.wikipedia.org/wiki/Systems_architecture) to conceptually model the [structure](https://en.wikipedia.org/wiki/Structure), [behavior](https://en.wikipedia.org/wiki/Behavior), and more [views](https://en.wikipedia.org/wiki/View_model) of a system.

The [Business Process Modeling Notation](https://en.wikipedia.org/wiki/Business_Process_Modeling_Notation) (BPMN), a graphical representation for specifying business processes in a workflow, can also be considered to be a systems modeling language.

* Software engineering occurs as a consequence of system engineering

System engineering may take on **2 different forms** depending on the application domain

**“Business process” engineering** – conducted when the context of the work focuses on a business enterprise

**Product engineering** – conducted when the context of the work focuses on a product that is to be built

Both forms bring order to the development of computer-based systems

Both forms work to allocate a role for computer software and to establish the links that tie software to other elements of a computer-based system

* System (Webster)
  + A set or arrangement of things so related as to form a unity or organic whole
  + A set of facts, principles, rules. etc., … to show a logical plan linking the various parts
  + A method or plan of classification or arrangement

An established way of doing something such as a method or procedure

**COMPUTER-BASED SYSTEM**

Defined: A set or arrangement of elements that are organized to accomplish some predefined goal by processing information

The goal may be to support some business function or to develop a product that can be sold to generate business revenue

A computer-based system makes use of system elements

Elements constituting one system may represent one macro element of a still larger system

Example :

* + A factory automation system may consist of a numerical control machine, robots, and data entry devices; each can be its own system
  + At the next lower hierarchical level, a manufacturing cell is its own computer-based system that may integrate other macro elements

The role of the system engineer is to define the elements of a specific computer-based system in the context of the overall hierarchy of systems

A computer-based system makes use of the following four system elements that combine in a variety of ways to transform information

**Software**: computer programs, data structures, and related work products that serve to effect the logical method, procedure, or control that is required

**Hardware**: electronic devices that provide computing capability, interconnectivity devices that enable flow of data, and electromechanical devices that provide external functions

**People**: Users and operators of hardware and software

**Database**: A large, organized collection of information that is accessed via software and persists over time

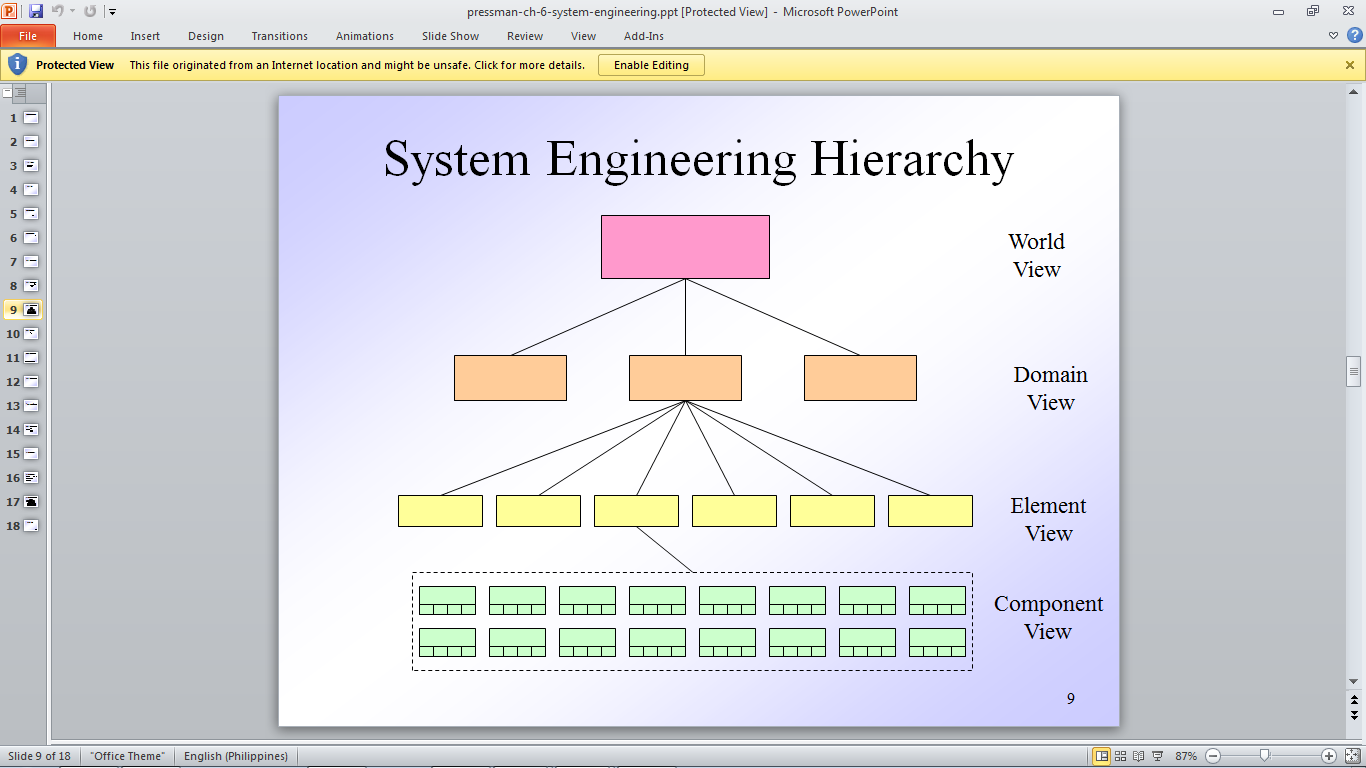
The uses of these elements are described in the following:

* + **Documentation**: Descriptive information that portrays the use and operation of the system
  + **Procedures**: The steps that define the specific use of each system element or the procedural context in which the system resides

**SOFTWARE ENGINEERING PROCESS**

1. The system engineering process begins with a world view; the business or product domain is examined to ensure that the proper business or technology context can be established
2. The world view is refined to focus on a specific domain of interest
3. Within a specific domain, the need for targeted system elements is analyzed
4. Finally, the analysis, design, and construction of a targeted system element are initiated
5. At the world view level, a very broad context is established
6. At the bottom level, detailed technical activities are conducted by the relevant engineering discipline (e.g., software engineering)

**SYSTEM ENGINEERING HIERARCHY**



**System Modeling (at each view level)**

* Defines the processes (e.g., domain classes in OO terminology) that serve the needs of the view under consideration
* Represents the behavior of the processes and the assumptions on which the behavior is based
* Explicitly defines intra-level and inter-level input that form links between entities in the model
* Represents all linkages (including output) that will enable the engineer to better understand the view
* May result in models that call for one of the following
  + Completely automated solution
  + A semi-automated solution
  + A non-automated (i.e., manual) approach

**Factors to Consider when Constructing a Model**

* Assumptions
  + These reduce the number of possible variations, thus enabling a model to reflect the problem in a reasonable manner
* Simplifications
  + These enable the model to be created in a timely manner
* Limitations
  + These help to bound the maximum and minimum values of the system
* Constraints
  + These guide the manner in which the model is created and the approach taken when the model is implemented
* Preferences
  + These indicate the preferred solution for all data, functions, and behavior
  + They are driven by customer requirements

**System Modeling with UML**

* The Uniform Modeling Language (UML) provides diagrams for analysis and design at both the system and software levels
* Examples
  + Use case diagrams
  + Activity diagrams
  + Class diagrams
  + State diagrams

**“Business Process” Engineering**

“Business process” engineering defines architectures that will enable a business to use information effectively

It involves the specification of the appropriate computing architecture and the development of the software architecture for the organization's computing resources

Three different architectures must be analyzed and designed within the context of business objectives and goals

* + The data architecture provides a framework for the information needs of a business (e.g., ERD)
  + The application architecture encompasses those elements of a system that transform objects within the data architecture for some business purpose
  + The technology infrastructure provides the foundation for the data and application architectures

It includes the hardware and software that are used to support the applications and data

**Product Engineering**

Product engineering translates the customer's desire for a set of defined capabilities into a working product

It achieves this goal by establishing a product architecture and a support infrastructure

* + Product architecture components consist of people, hardware, software, and data
  + Support infrastructure includes the technology required to tie the components together and the information to support the components

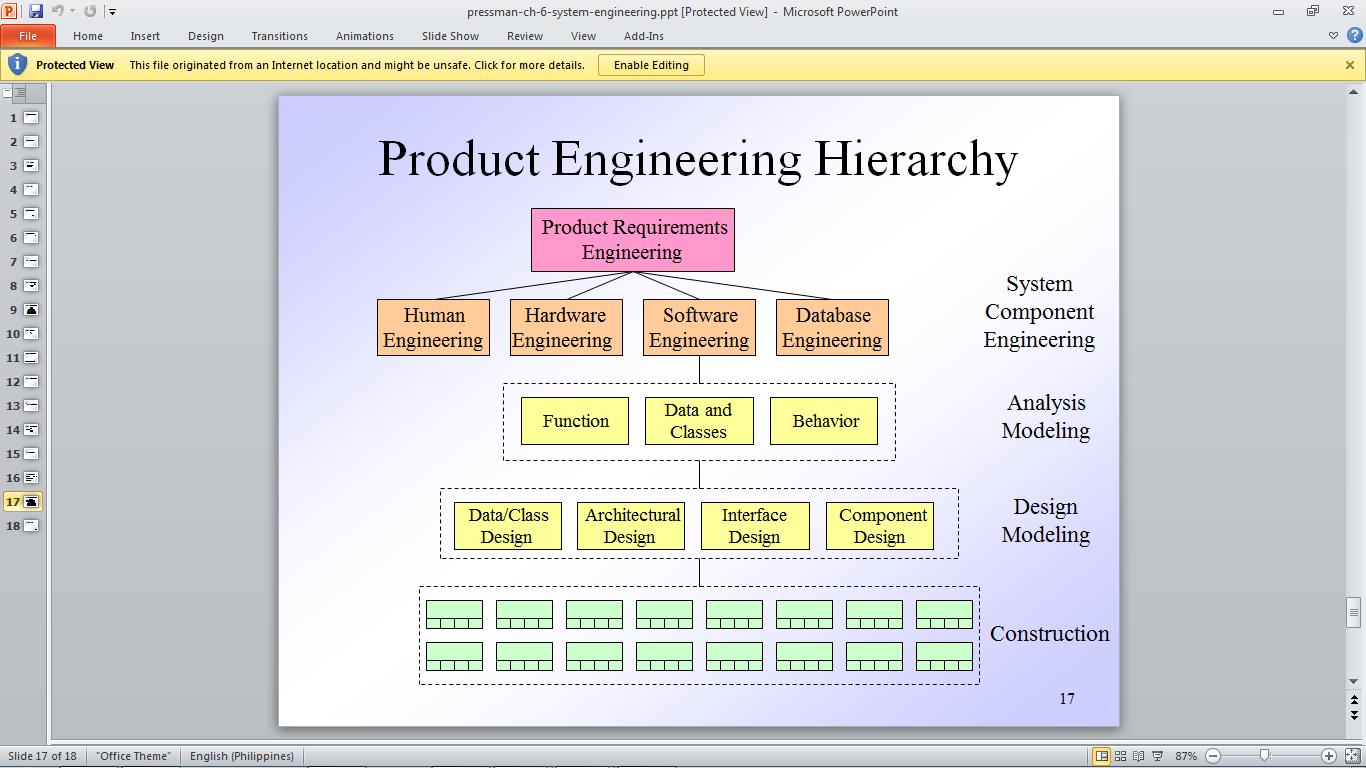
Requirements engineering elicits the requirements from the customer and allocates function and behavior to each of the four components

System component engineering happens next as a set of concurrent activities that address each of the components separately

* + Each component takes a domain-specific view but maintains communication with the other domains
  + The actual activities of the engineering discipline takes on an element view

Analysis modeling allocates requirements into function, data, and behavior

Design modeling maps the analysis model into data/class, architectural, interface, and component design



**SOFTWARE PROTOTYPING**

what is software prototyping ?

It is the process of implementing the presumed software requirements with an intention to learn more about the actual requirements or alternative design that satisfies the actual set of requirements.

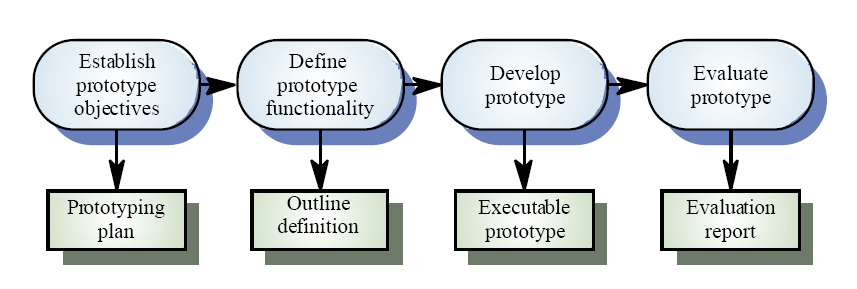
Need for software prototyping

-To assess the set of requirements that makes a product successful in the market

-To test the feasibility without building the whole system.

-To make end-user involved in the design phase

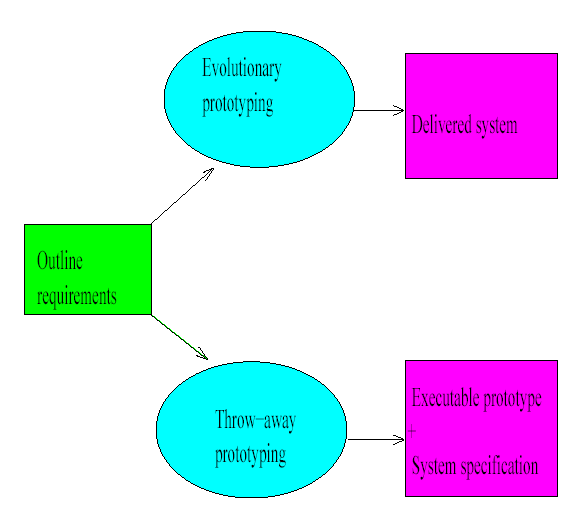
**Phases of prototyping**



**Different Phases**

* Define the prototype objectives
* Define the functionality
* Develop Prototype
* Evaluate Prototype.

**Types of Prototyping**



* Throw away prototyping
* Evolutionary prototyping
* Operational prototyping
* **Throw away prototyping**

Objective : Derive end system requirements

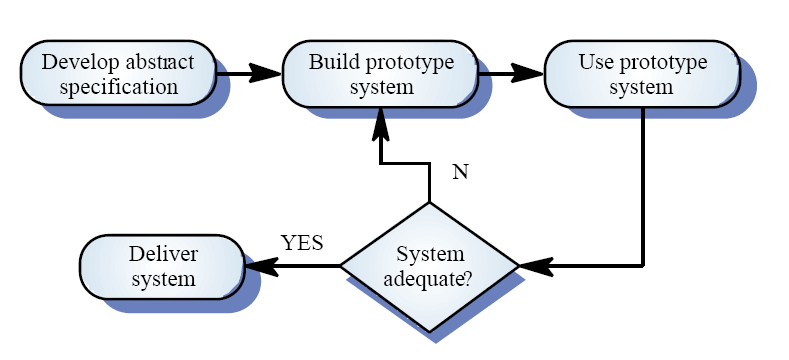
**Throw away prototyping** is one type of approach where an initial prototype is built mainly focusing on the poorly understood requirements

Once the requirements are understood requirements document is updated and a conventional development process is followed to build system.

* **Evolutionary Prototyping**

Objective: Deliver a working system + requirements

**Evolutionary prototyping** is the one in which a system is build using the well understood requirements.



* Advantages –
  + - Accelerated Delivery
    - Makes User Commit
    - Look like feel
* Disadvantages –
  + - Availability of specialist skills
    - Maintenance over long term
* **Operational Prototyping**

**Operational Prototyping** used when requirements are either critical and understood or not critical and poorly understood. Throw away prototypes are selectively built on top of evolutionary prototype. A trained prototype keeps track of user .

**TOOLS AND TECHNIQUES**

* Low level tools
* High Level languages
* Fourth Generation Languages (4GL)
* Visual programming

**Benefits of Software Prototyping**

* It makes the developers clear about the missing requirements. Let’s the developers know what actually the users want.
* Reduces the loss by bringing the manufacturer to a conclusion weather the system which we are about to build is feasible or not rather than building the whole system and finding it.
* One can have a working system in beforehand.
* It brings the user to get involved in the system design

**Troubles of Software Prototyping**

* Developers may lose the focus on real purpose of prototype and comprise with the quality of system.
* New born ideas will be plundered at the initial stages
* Prototyping will not reveal the non-functional requirements like robustness, safety etc .

**SOFTWARE REQUIREMENT SPECIFICATION [SRS]**

A **software requirements specification (SRS)** is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

* Correct
* Unambiguous
* Complete
* Consistent
* Ranked for importance and/or stability
* Verifiable
* Modifiable
* Traceable

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.

