|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Key Concepts** | **Explore concepts' significance and relevance** | **Establish relevance, make sense and meaning -Find real-life contexts** | **Establish relevance, make sense and meaning -Find interdisciplinary connections** | **Engage in critical thinking** | **Technology, tools and techniques** | **Plan project management** | **Project specification and sketch** |
| Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.  The various kinds of designs for a system include :-  **1.** Architectural design.  **2.** Logical design.  **3.** Physical design.  Alternative design methodologies include:-  **1.** Rapid application development  (RAD).  **2.** Joint application development  (JAD). | System designing in terms of software engineering has its own value and importance in the system development process as a whole. To mention it may though seem as simple as anything or simply the design of systems, but in a broader sense it implies a systematic and rigorous approach to design such a system which fulfils all the practical aspects including flexibility, efficiency and security.  For the small programs we tend to implement as part of a semester project, or the simple “one-off/get-it-done” programs at work, system design rarely plays a part. However, even in the smallest problems a top-level system design is critical for consistency and ease of use. Class components, regardless of how precise and accurate they are in their own internal design, if they aren’t externally consistent with other objects, the system will be brittle, and difficult to use. | Design teams in automotive, transportation, civil aviation, and the like, which demand that each element in the design and the test bench be traceable to its governing element in the requirements document implement designing in real world. These design teams can identify exactly which parts of the design will require testing and, possibly, alteration in order to comply with a changed requirement. And they can say which modules will have to change in the test bench.  But in much of the design world, formal requirements traceability is pure fantasy. Traceability on such projects exists only in the memories of the design team members. Our ideal scenario plays out for these real-world situations. | Related disciplines include:-  **1.** [**Benchmarking**](https://en.wikipedia.org/wiki/Benchmarking)– is an effort to evaluate how current systems perform.  2. [**Computer programming**](https://en.wikipedia.org/wiki/Computer_programming)**and**[**debugging**](https://en.wikipedia.org/wiki/Debugging) **–** in the software world, or detailed design in the consumer, enterprise or commercial world - specifies the final system components.  **3.** [**Design**](https://en.wikipedia.org/wiki/Design)**–** designers will produce one or more '[models](https://en.wikipedia.org/wiki/Systems_modeling)' of what they see a system eventually looking like, with ideas from the analysis section either used or discarded.  **4.** [**Requirements analysis**](https://en.wikipedia.org/wiki/Requirements_analysis)**–** analyses the needs of the end users or customers.  **5.** [**Systems architecture**](https://en.wikipedia.org/wiki/Systems_architecture)**–** creates a [blueprint](https://en.wikipedia.org/wiki/Blueprint) for the design with the necessary structure and behaviour specifications for the hardware, software, people and data resources. In many cases, multiple architectures are evaluated before one is selected.  **5.** [**System testing**](https://en.wikipedia.org/wiki/System_testing)**–** evaluates the system's actual functionality in relation to expected or intended functionality, including all integration aspects. | Critical thinking involves looking deeper into the methodology being selected, figuring out the type of cohesion, which is functional in our case, and coupling, which is data in our case.  Concurrency, modularization, software design level (Architectural, High-level and Detailed) and design verification play and important part. | Techniques for Designing Software Structure:-  **1. Structure Charts** – It is a graphic representation of a hierarchy of modules and the relationships between them.  **2. Data Flow Oriented Design –** Data flow oriented decomposition is a method of deriving the modular design based on the data flow diagram.  **3. Data Structure Oriented Design –**  Data Structure Oriented Design is a technique of modular decomposition which transforms a representation of a data structure into a representation of software.  **4. Object-Oriented Design** **–** Object  Oriented Design is an approach to modular decomposition which views a problem in terms of objects and operations and defines the solution in the same terms.  **5. Pseudo-Code –**  Pseudo code may be considered as augmented programming language, full of comments and descriptions. | The system design document is being worked upon. We have decided to learn while making the actual document, and not just oral meetings.  Immediate deliverables are nothing yet, but in the future, SDD (Systems Design Document) has to be delivered.  PISE-PBL Sub Project 4 will be form September 29-  October 2. | As mentioned above.  ---------do-------- |