Week 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Start Time | End Time | Total Time | Particulars |
| 8/1/2020 | 11:14AM |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Software Design Introduction:**

Software design is what we call the deliverable, design, the noun, and what we call the process to make that design.

Design, the verb, is the creative process of transforming the problem into a solution. In our case, transforming a requirement specification into a detailed description of the software that's code-ready.

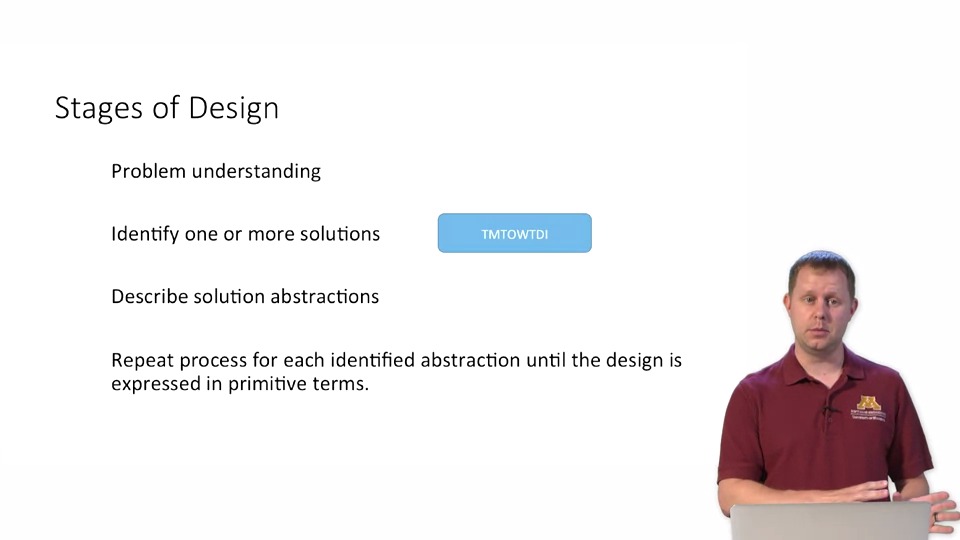
Architecture vs Design:

Architecture:

Large scale decision like

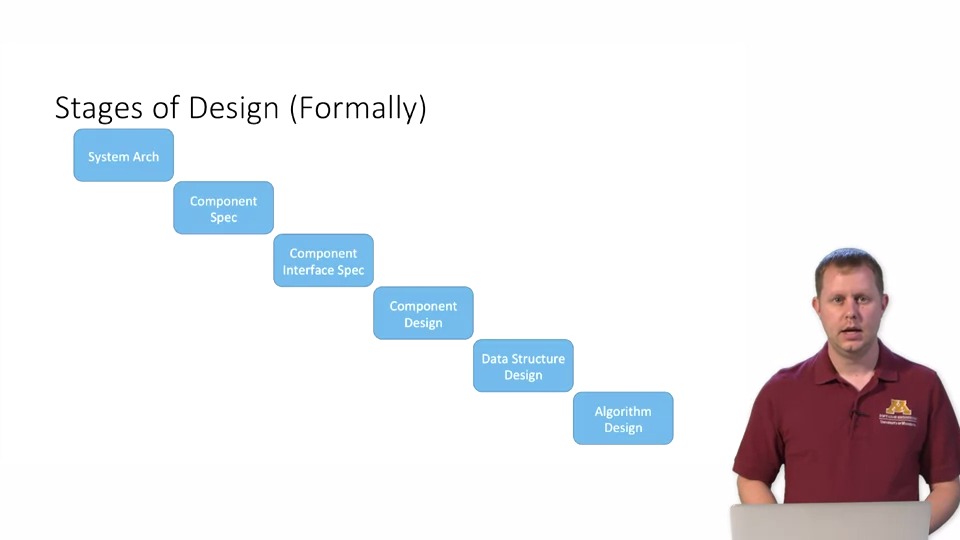
1. Buy or build software
2. Security
3. Appointing Resources
4. Fund related
5. Cost

Stages of Design:



TMTOWTDI: There’s more than one way to do it. There is almost always another way to reach the same singular goal, so consider multiple alternatives before deciding definitively which one to pursue.

Formal Stages of Design:



Architecture: Separate behaviour responsibility into components and, determine how those components will interact through interfaces.

Design: Design individual components in isolation.

**Software Design Modularity:**

Aspects of Modularity:

1. Coupling:

Defines how well does a module work together.

1. Cohesion:

Defines how well a module meets a single well defined goal.

1. Information Hiding
2. Information hiding describes our ability to abstract away information and knowledge in a way that allows us to complete complex work in parallel without having to know all the implementation details concerning how the task will be completed eventually.
3. Basically we know, what a module is doing and not how.
4. Data Encapsulation
5. data encapsulation refers to the idea that we can contain constructs and concepts within a module, allowing us to much more easily understand and manipulate the concept when we're looking at it in relative isolation.
6. It ensures protecting data from unauthorised access and maintaining integrity.
7. Only developer can modify the data.
8. And if in future we know that data is corrupted, it will happen only inside the module.
9. It makes software robust because later we can upgrade the module and

Coupling, cohesion are measures of how well modules work together and how well each individual module meets a certain single well-defined task and they tend to go together

Primary Goals of Modularity:

1. Decomposability
2. Composability
3. Ease of Understanding