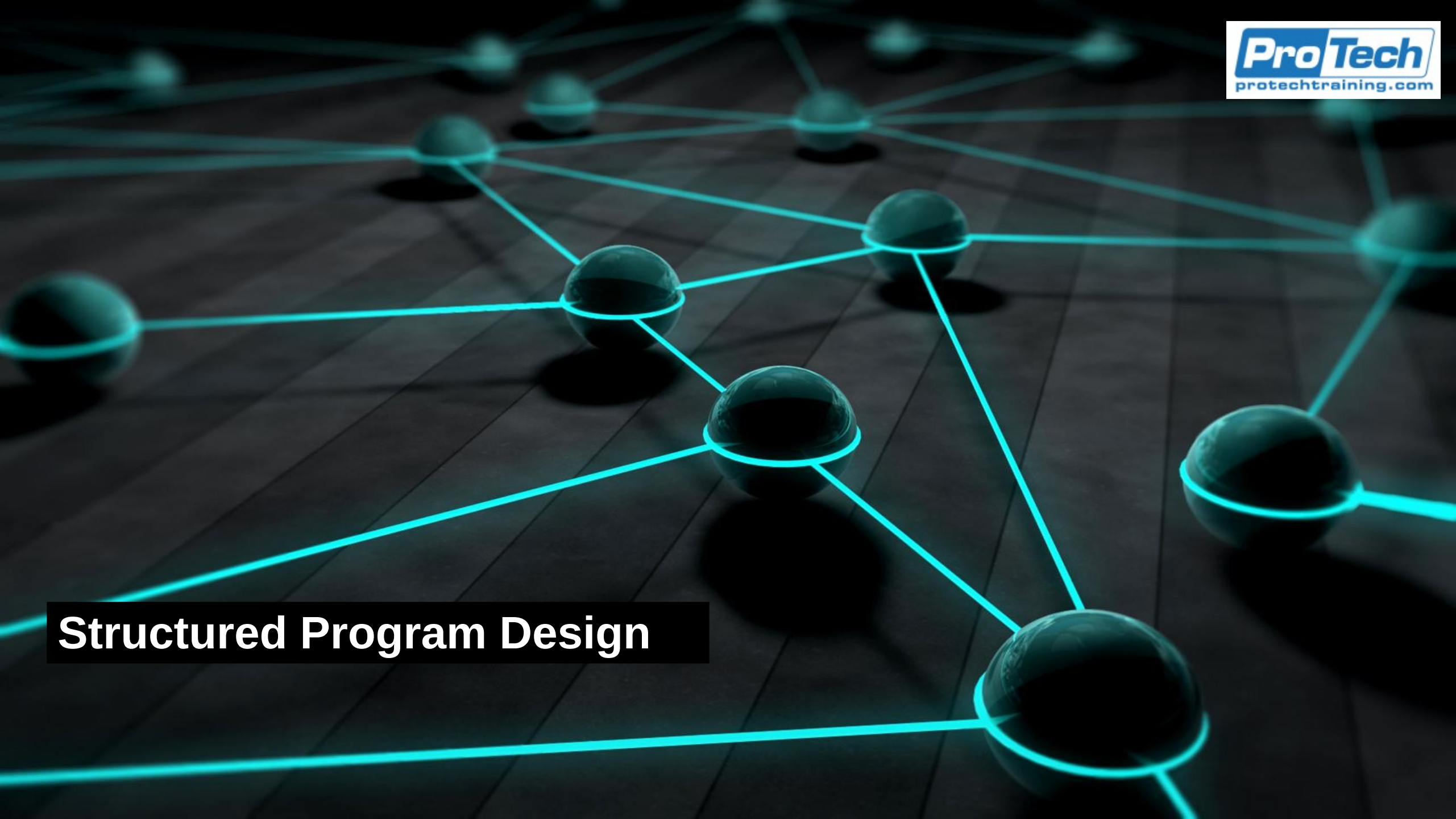


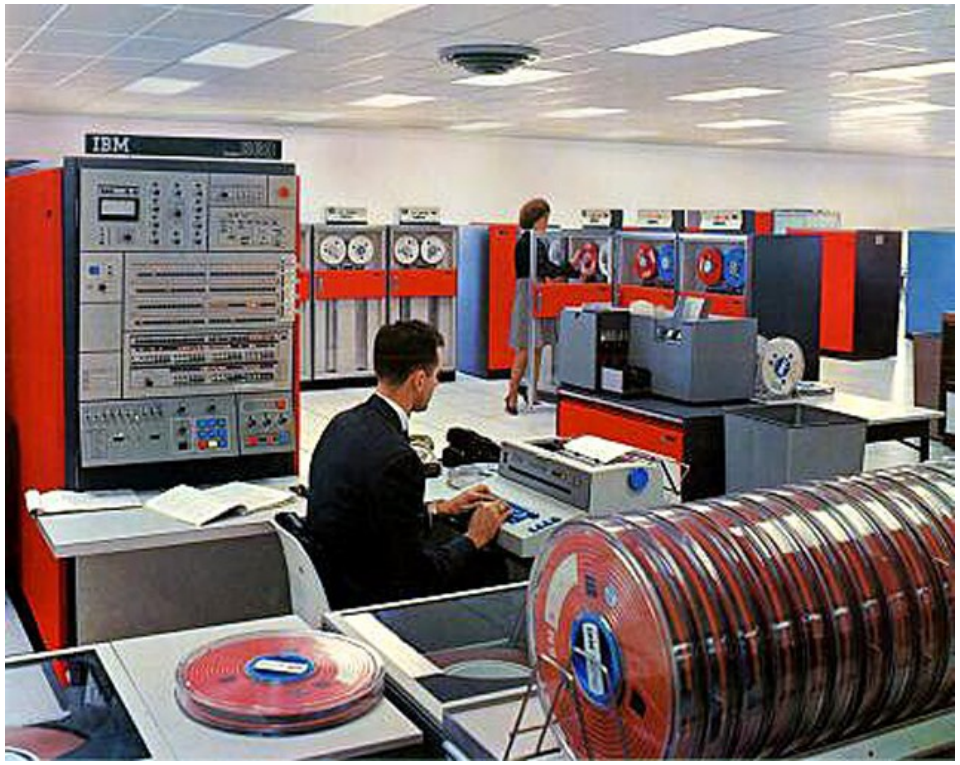
Structured Program Design

Structured Programming

- Structured programming is procedural programming
  - That means we generally write imperative style code
  - Imperative code is organized in procedures which can be functions or subroutines
- Not all procedural code is structured code
  - Structured programming is writing procedural code based on a set of structured best practices
  - The rise of software engineering let to developing structured programming
  - Much of the principles were based rigorous mathematical computing theory
    - Looked for ways to write code that satisfied many of the engineering principles discussed earlier
  - Led to the development of structured analysis and design (SA&D)
    - Analytical techniques for analyzing a problem domain
    - Design techniques that resulted in designs that could be implemented easily in structured code

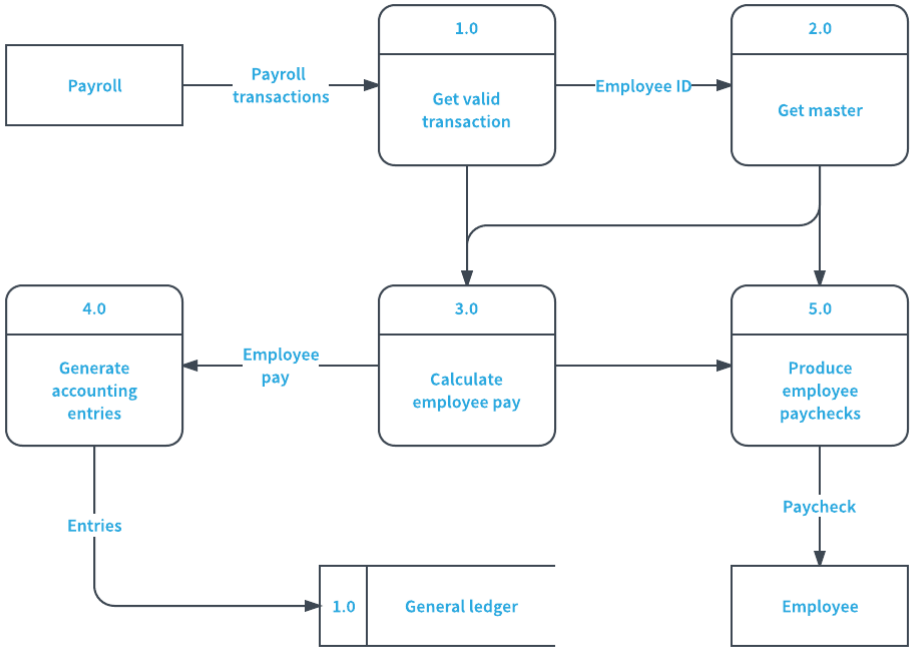
The Basic Problem

- SP was developed when almost all computing was batch computing
  - A set of data would be read in as input
  - The data would be processed by a series of procedures
  - Each procedure would either modify the data or create new data from it (transformations and reporting)
  - The modified data or reports would become the output of the system
- Challenges
  - The data sets could be quite large
  - The processing could be complex, often automating complex business processes
  - For example, processing a day's transactions at a bank
    - Prints statements, updates accounts, etc.



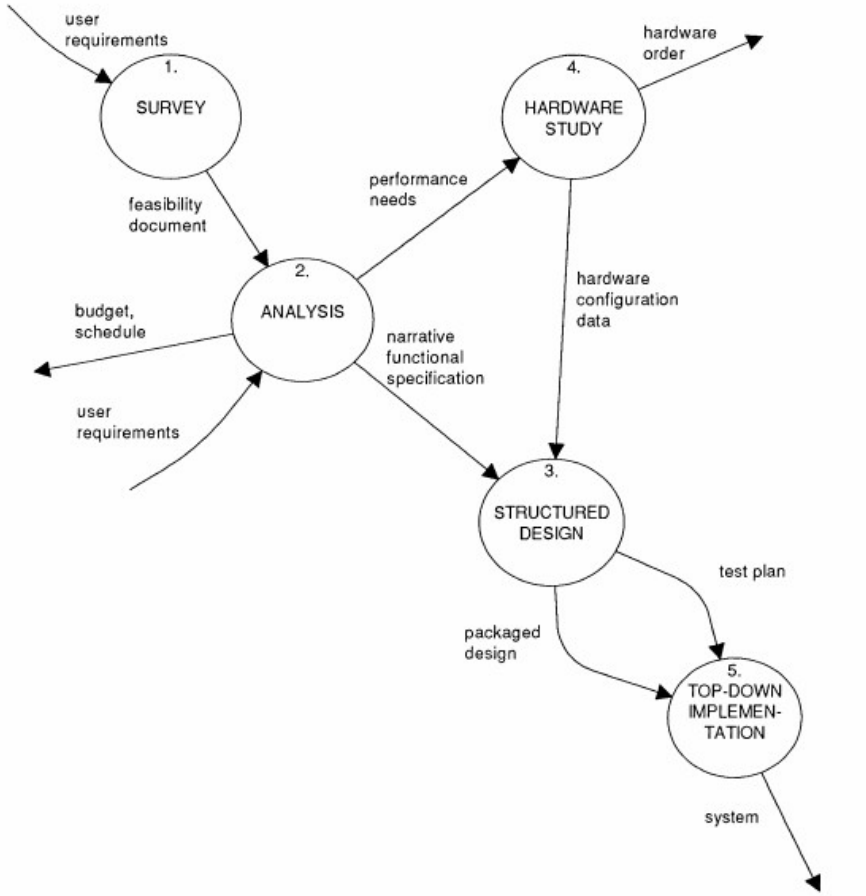
Data Flows

- One of the primary concerns of designing code was to understand how data flowed through the system
  - This was documented as a data flow diagram or DFD
  - Identified the inputs and outputs of the system
  - And all the processes that transformed the data
  - Documents all possible flows through the system
- A DFD does not document
  - The logic used in the transformations
  - The structure of the data being processed



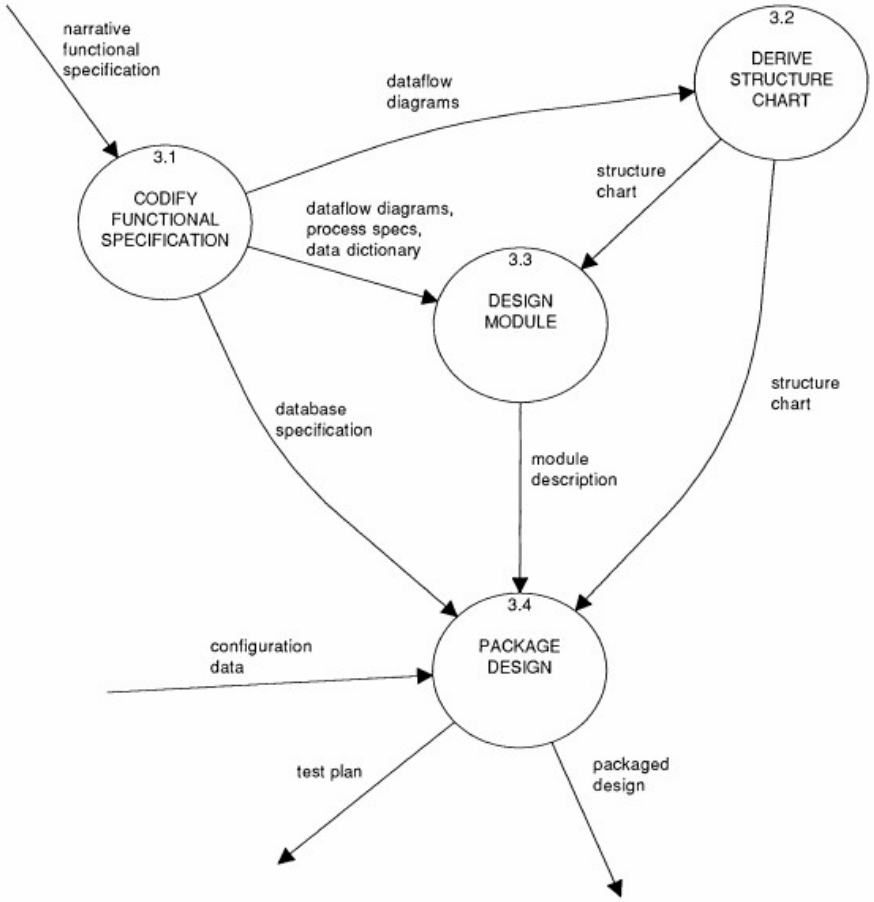
Data Flows

- Often developed by duplicating the manual flow of data (documents) in a manual system
  - The logic used in the transformations
- Often “leveled”
  - That means that a process could be broken down in to levles of sub-processes
  - The sub-processes inside a process have their own DFD
- For example
  - The DFD on the right shows the flow of data for a design process.
  - Process 3 is the structured design process



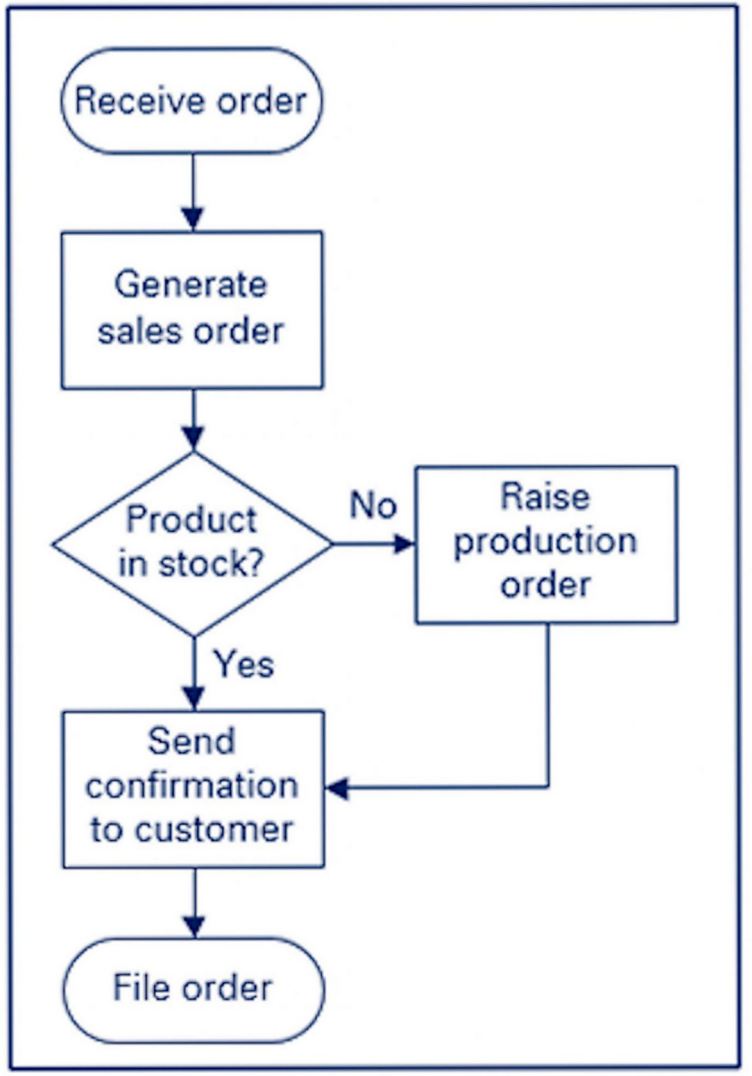
Data Flows

- Leveling
  - The DFD on the right is a look inside the process labeled as 3. Structured Design
- There are 4 sub-processes
  - These are at level 2.
  - They are labeled as 3.1, 3.2, etc
  - This shows they are inside the process bubble number 3 at the higher level



Flowcharts

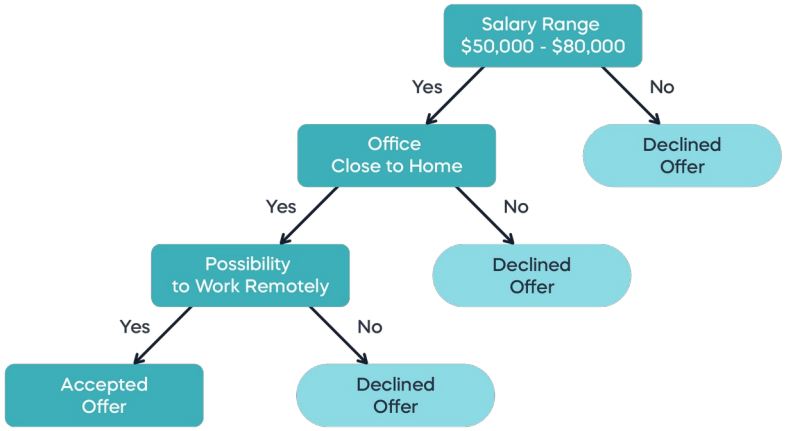
- The actual logic used to do the processing in a process is represented by a flowchart.
  - Shows a set of steps and decision used to make decisions
  - Not intended to show data flows, but does show steps that are executed during a decision
- Derived from an electrical circuit diagram
  - Logic used to be hard coded in early computers at the hardware level
  - For smaller applications, also functioned as a DFD
  - Software engineering uses the flowchart only for visually depicting logic
  - This allows a simple translation into procedural code;



Flowchart Alternative

- Flowcharts are not the only logic tools
- Also used are:
  - Structured English (restricted natural language)
  - Decision tables
  - Decision trees

		Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8	Rule 9
Conditions	Email	Empty	Empty	Empty	Invalid	Invalid	Invalid	Valid	Valid	Valid
	Password	Empty	Invalid	Valid	Empty	Invalid	Valid	Empty	Invalid	Valid
Actions	Authorization	No	No	No	No	No	No	No	No	Yes



Data Dictionary

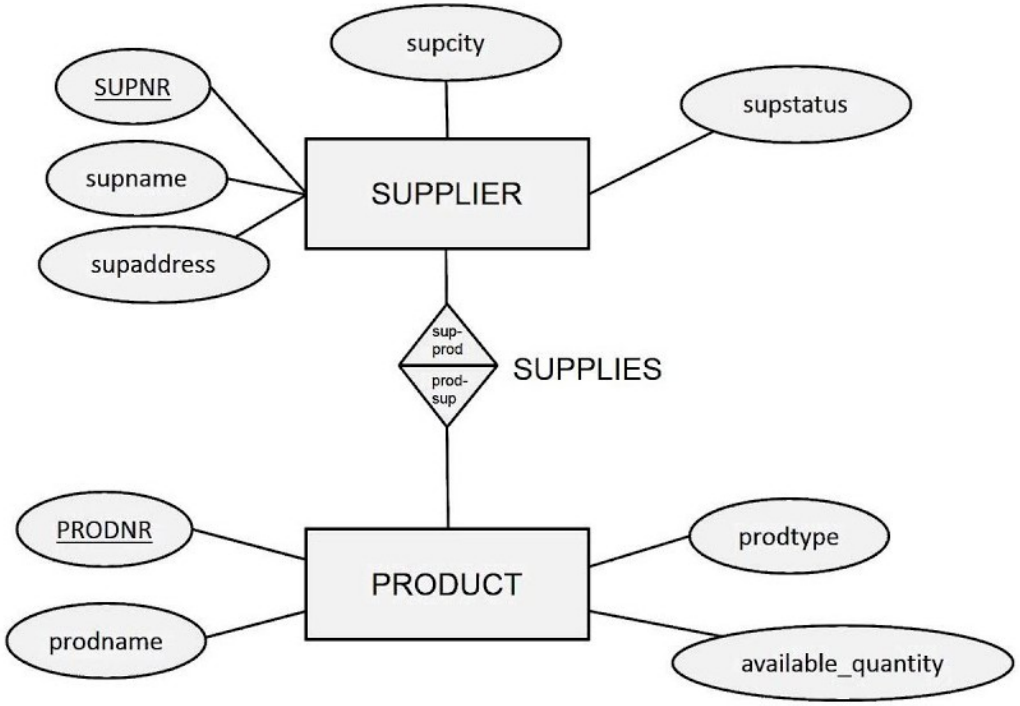
- It explicitly defines the data used in a system
  - Not a relational model – not defined in terms of tables
  - Provides a description of the data in a data record – data sets are made of of individual data records
  - And for each record, specifies the data fields, data types and relations with other data records
  - These were often implemented in hierarchical or network data bases which are not longer used
- Uses of the data dictionary
  - Defines the layout and representation of the content of data record and dependencies between records
  - Goal was to support code that processed a specific kind of data organized in a specific format

Data Dictionary: Student Details

	Field Name	Data Type	Data Format	Field Size	Description	Example
PK	StudentID	Text	XNNNNNNN	8	Unique Identification for each Student	S1234567
	First_Name	Text		20	Student's First Name	Connor
	Last_Name	Text		20	Student's Surname	Thomas
	Year_Group	Number	##	2	Year level the student is in	8
	Date_of_Birth	Date/Time	DD/MM/YYYY	10	Date the Student was born	14/08/2005
	Student_Image	Image	.jpg	---	Profile photo of the Student	---
	School_Team	Text		10	Coloured team student assigned	Blue

Entity Relations

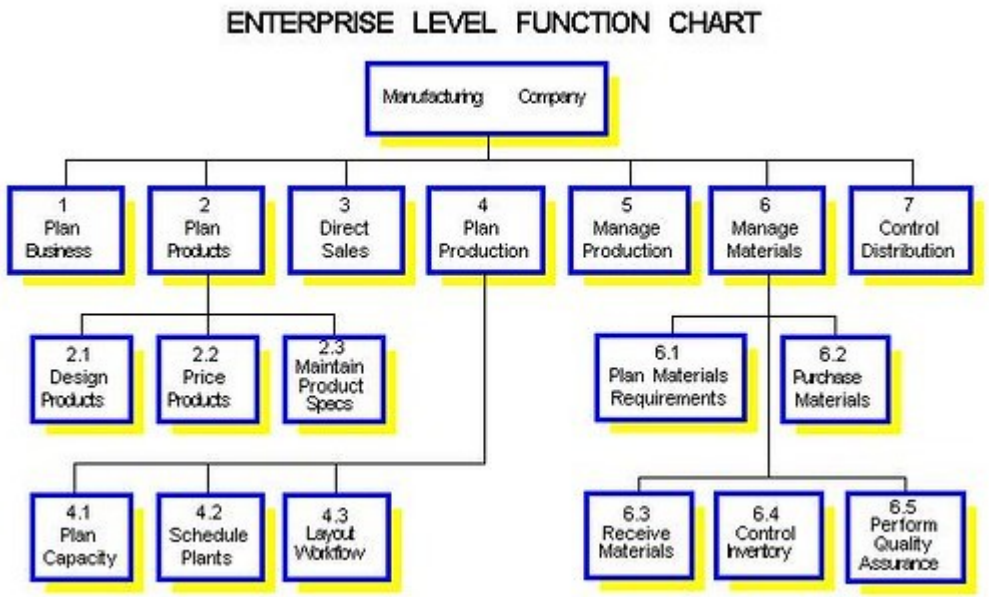
- Often visually described in an entity relationship model and diagram
  - We will deal with the model part in week 4.
  - A typical ER diagram is shown
  - Entities are rectangles, relationships are diamonds
  - Attribute of an entity are ovals
- Modeling of the data is done at the logical level
  - Should be implementable in a variety of different structures
  - For example, dimensional model, relational model or a NOSQL structure

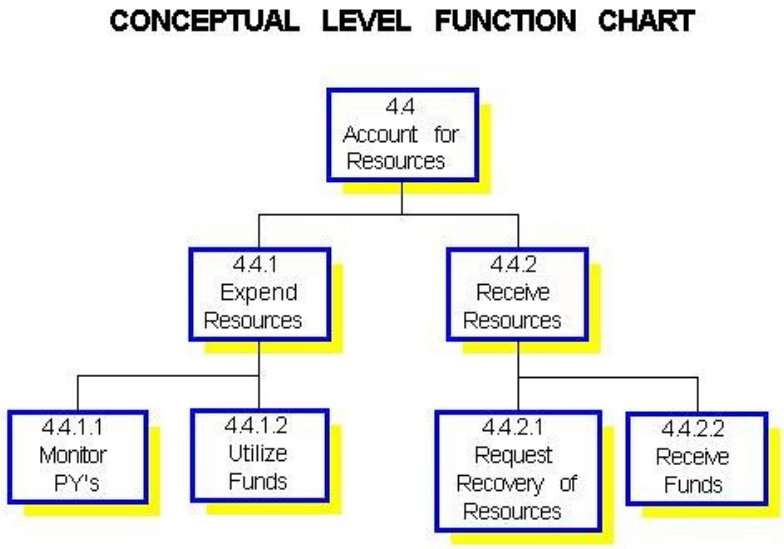


Structural Decomposition

- A top-down, model-driven technique for understanding and specifying systems.
  - It emerged in the 1970s to help engineers design large, reliable information systems before coding.
  - Breaks a system into smaller, well-defined parts (structured decomposition)
  - Then model the data and processes with the diagrams and processes we have just seen.
- Functional decomposition
  - Breaking down a complex system into smaller, more manageable subsystems or functions
  - Continue until each piece is simple enough to understand and design.
- Start with the overall system function
  - Split into major processes as cohesive and loosely coupled modules
  - Split those modules sub-modules.
  - Continue until each module is small, cohesive, and has a single purpose.
  - We follow engineering design concepts at each step
- Helps manage complexity, makes it easier to understand, and simplifies testing.

Structural Decomposition

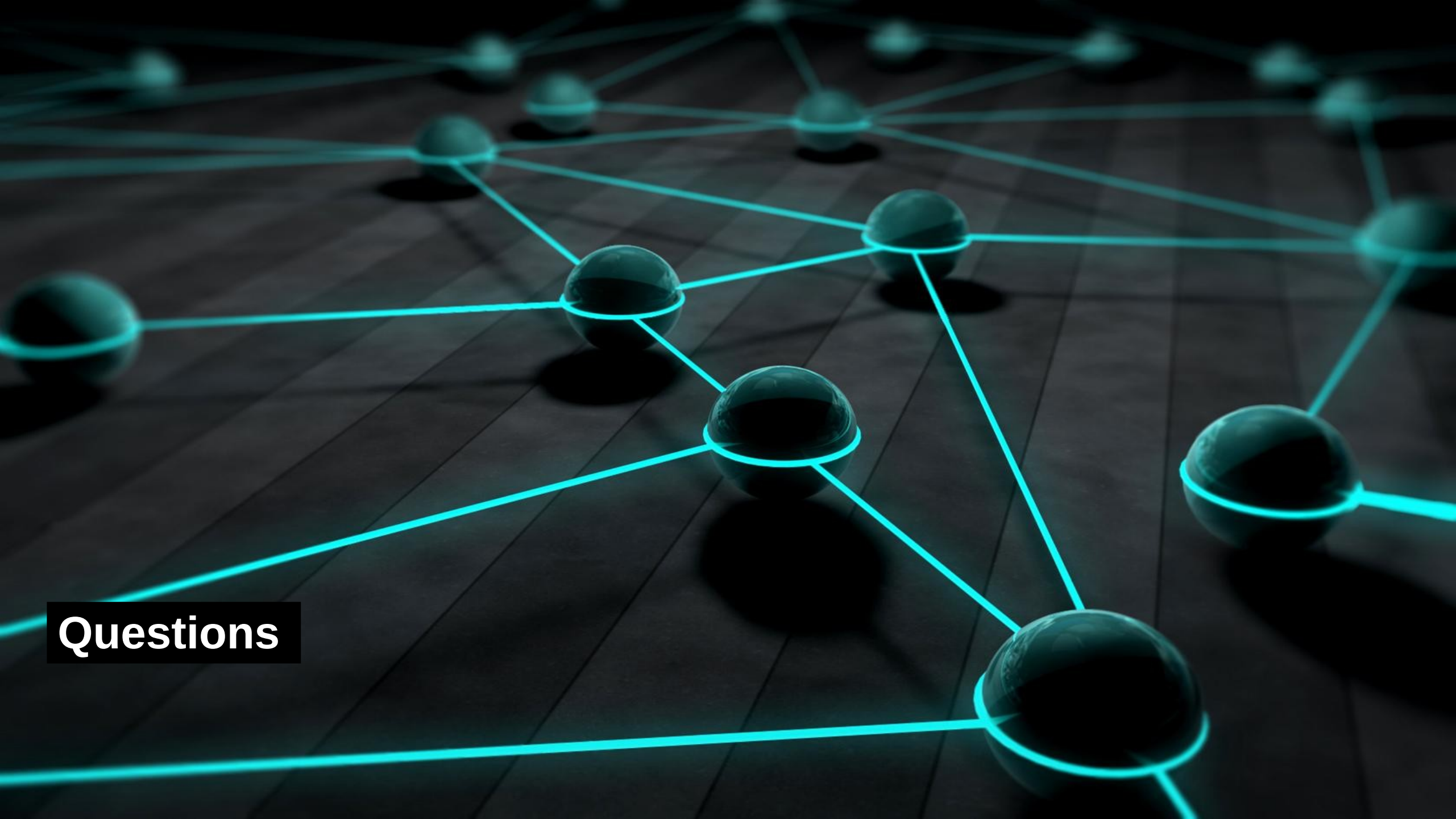




- Since most applications are made of of layers
  - A functional decomposition is a depiction of layers
  - Like the functions of departments in a company
  - Or the allocation of specific tasks in a business process

Importance of Structured Programming

- Many of the early programming languages used the structured approach
  - Later OO and other programming languages build on top of the structured approach
  - The code inside class methods is, for example, all structured code.
  - The code inside functional programming functions is also structured code
- The approach was successful
  - It allowed for massive amounts of computerization starting in the 1960s of systems that are still mission critical for large organizations today
  - A lot of corporate data processing still reading and processing data in batch mode
- Modern languages
  - Many modern languages like GO and Rust are basically structured programming languages
    - They add OO and functional capabilities, as well as more modern features like support for concurrent development
    - For a lot of tasks, like systems programming, structured procedural code often is the leanest and most efficient way to code.


Questions