IF3141 SISTEM INFORMASI

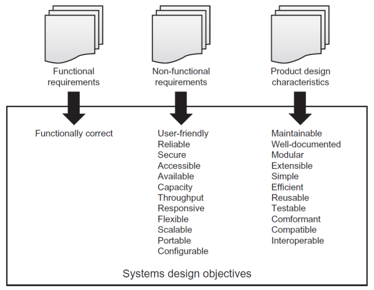
System Design

Semester I 2023/2024

1

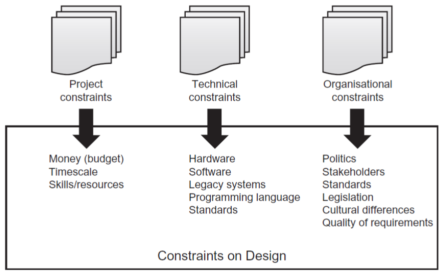
Objective of Systems Design

...to formulate a solution to realise the system requirements...



2

Constraints Upon Systems Design

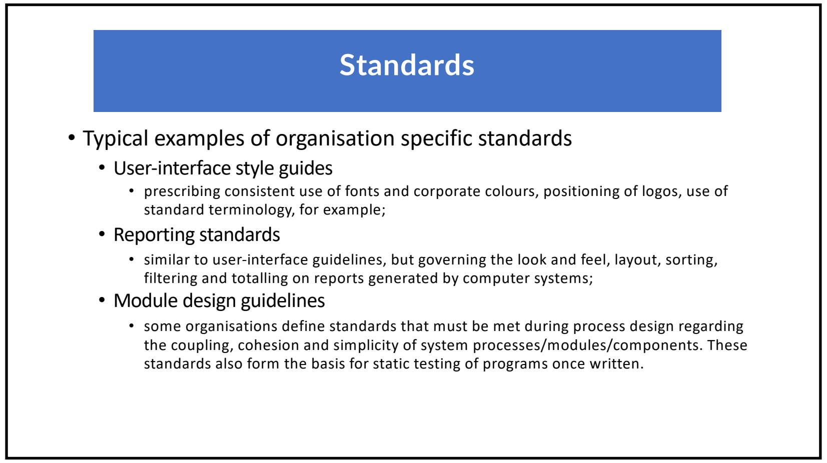


3

Standards

- **Typical industry-wide technical standards that must be complied :**
  - User-interface style guides.
  - Data formats
    such as the use of open standards like XML.
  - Protocols
    such as SOAP (Simple Object Access Protocol) and REST (Representational State Transfer), used for passing messages in a service-oriented architecture.
  - Security
    such as ISO 27001 (the international standard describing best practice for an Information Security Management System), ISO/IEC 7498 (Open Systems Interconnect (OSI) security model), ISO/IEC 13335 (IT security management), ISO/IEC 27005 (Information technology – Security techniques – Information security risk management).
  - Application specific standards
    such as PCI DSS (Payment Card Industry Data Security Standard).
  - Modelling standards
    such as the use of the Unified Modeling Language (UML) notation when producing design models.

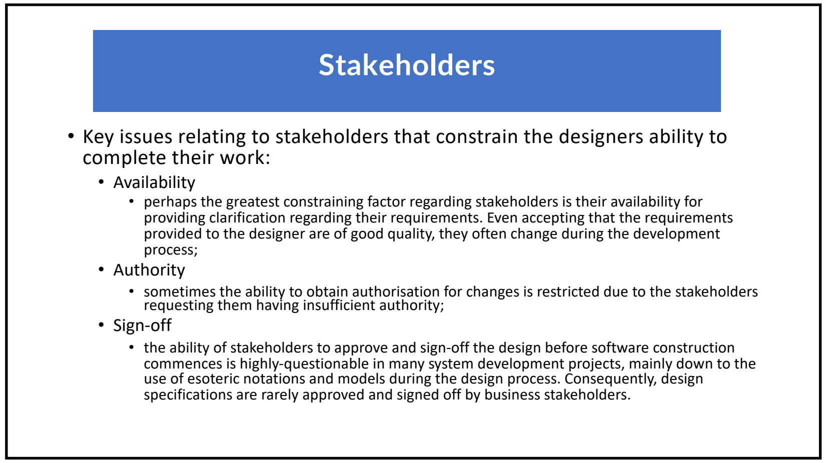
4



Standards

- Typical examples of organisation specific standards
  - User-interface style guides
    - prescribing consistent use of fonts and corporate colours, positioning of logos, use of standard terminology, for example;
  - Reporting standards
    - similar to user-interface guidelines, but governing the look and feel, layout, sorting, filtering and totalling on reports generated by computer systems;
  - Module design guidelines
    - some organisations define standards that must be met during process design regarding the coupling, cohesion and simplicity of system processes/modules/components. These standards also form the basis for static testing of programs once written.

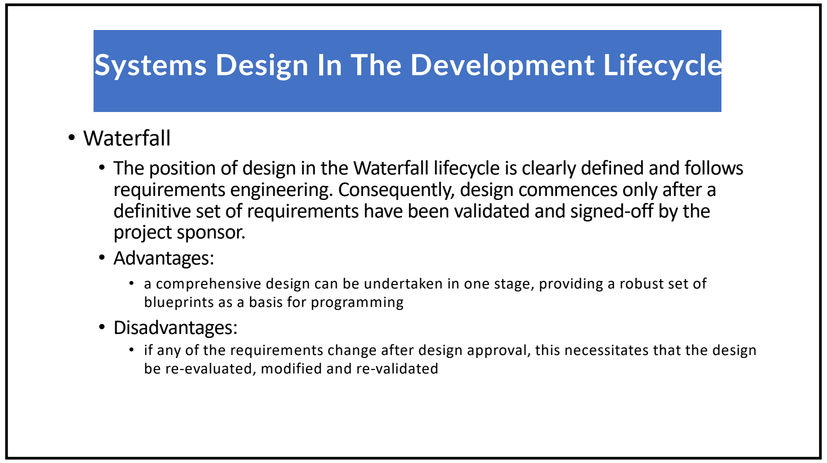
5



Stakeholders

- Key issues relating to stakeholders that constrain the designers ability to complete their work:
  - Availability
    - perhaps the greatest constraining factor regarding stakeholders is their availability for providing clarification regarding their requirements. Even accepting that the requirements provided to the designer are of good quality, they often change during the development process;
  - Authority
    - sometimes the ability to obtain authorisation for changes is restricted due to the stakeholders requesting them having insufficient authority;
  - Sign-off
    - the ability of stakeholders to approve and sign-off the design before software construction commences is highly-questionable in many system development projects, mainly down to the use of esoteric notations and models during the design process. Consequently, design specifications are rarely approved and signed off by business stakeholders.

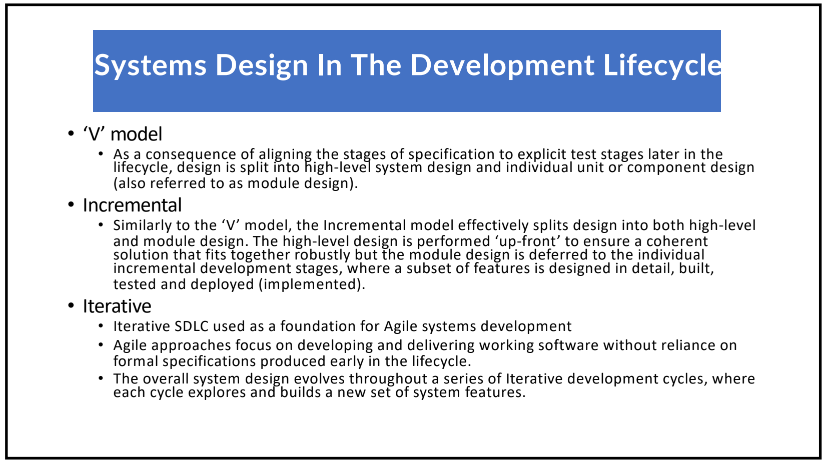
6



Systems Design In The Development Lifecycle

- Waterfall
  - The position of design in the Waterfall lifecycle is clearly defined and follows requirements engineering. Consequently, design commences only after a definitive set of requirements have been validated and signed-off by the project sponsor.
  - Advantages:
    - a comprehensive design can be undertaken in one stage, providing a robust set of blueprints as a basis for programming
  - Disadvantages:
    - if any of the requirements change after design approval, this necessitates that the design be re-evaluated, modified and re-validated

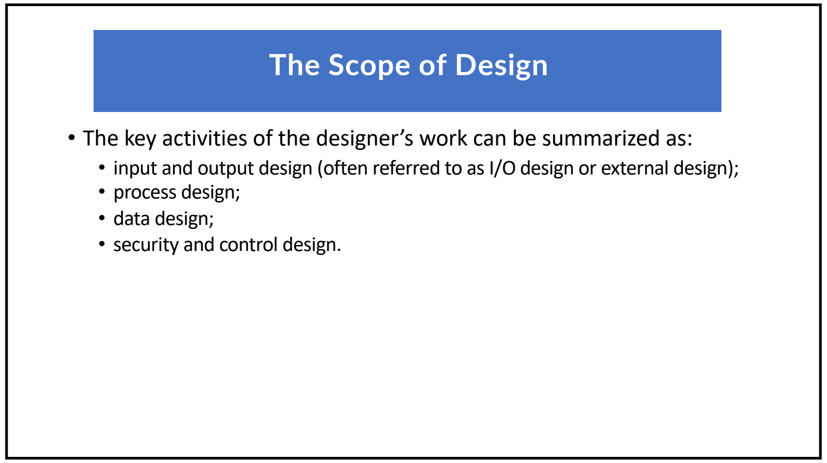
7



Systems Design In The Development Lifecycle

- 'V' model
  - As a consequence of aligning the stages of specification to explicit test stages later in the lifecycle, design is split into high-level system design and individual unit or component design (also referred to as module design).
- Incremental
  - Similarly to the 'V' model, the Incremental model effectively splits design into both high-level and module design. The high-level design is performed 'up-front' to ensure a coherent solution that fits together robustly but the module design is deferred to the individual incremental development stages, where a subset of features is designed in detail, built, tested and deployed (implemented).
- Iterative
  - Iterative SDLC used as a foundation for Agile systems development
  - Agile approaches focus on developing and delivering working software without reliance on formal specifications produced early in the lifecycle.
  - The overall system design evolves throughout a series of Iterative development cycles, where each cycle explores and builds a new set of system features.

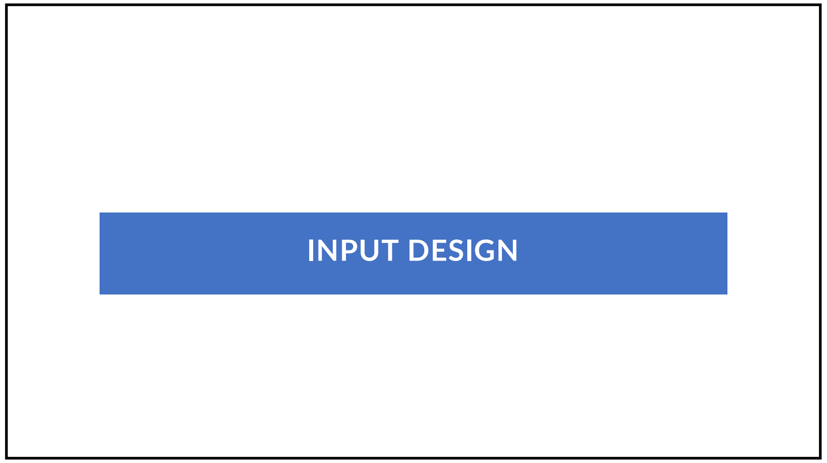
8



The Scope of Design

- The key activities of the designer’s work can be summarized as:
  - input and output design (often referred to as I/O design or external design);
  - process design;
  - data design;
  - security and control design.

9



INPUT DESIGN

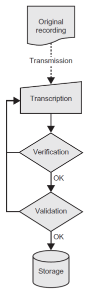
10

The Scope of Design: Input Design

- Input requirements are defined during requirements engineering and specified in either a requirements catalogue, a functional specification, or both.
- The functional specification and requirements catalogue are often supplemented by protot
- Input Design
  - The principal objective of input design is to specify how the system will collect data in its raw form (for example a paper-based form or a third-party system's database) and convert that data into an internal format (all computers use a binary code internally, which uses only the two binary digits 0 and 1) that the system can understand

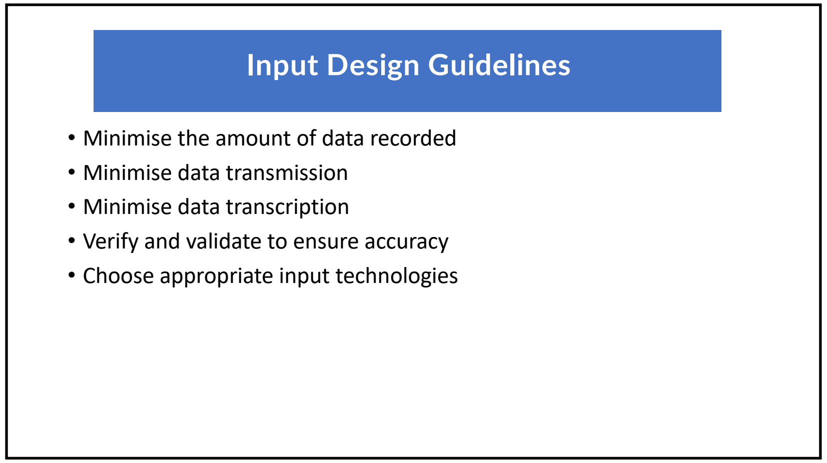
11

Stages During Data Input



- Original recording
  - Data is created or captured at its source.
- Transmission
  - The captured data is sent to some location, where it is converted into an internal format understandable by the system.
- Transcription
  - The captured data is converted (transcribed) into the internal format used by the computer system.
- Verification
  - The transcribed data is checked to ensure that no errors are introduced, most commonly, that the data entry operator has mis-keyed some of the data.
- Validation
  - The transcribed data is subjected to further checks to ensure that it complies with pre-defined data integrity principles and business rules defined by the business stakeholders.
- Storage
  - The final stage is to store the data in a permanent or semi-permanent storage system for future use.

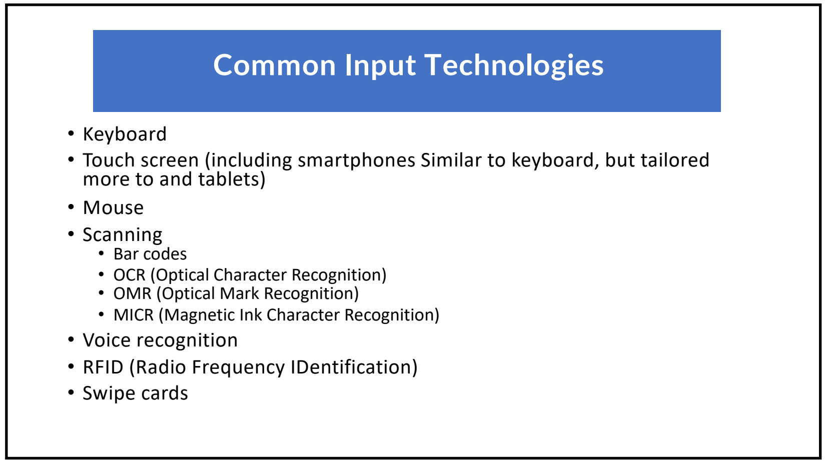
12



Input Design Guidelines

- Minimise the amount of data recorded
- Minimise data transmission
- Minimise data transcription
- Verify and validate to ensure accuracy
- Choose appropriate input technologies

13



Common Input Technologies

- Keyboard
- Touch screen (including smartphones Similar to keyboard, but tailored more to and tablets)
- Mouse
- Scanning
  - Bar codes
  - OCR (Optical Character Recognition)
  - OMR (Optical Mark Recognition)
  - MICR (Magnetic Ink Character Recognition)
- Voice recognition
- RFID (Radio Frequency IDentification)
- Swipe cards

14

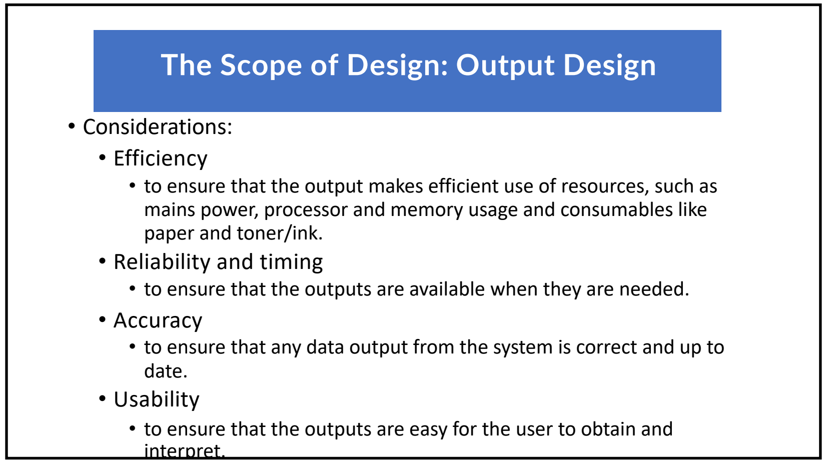
Input User-interfaces

- Menus
- Form filling
- Command language
- Question and answer

15

OUTPUT DESIGN

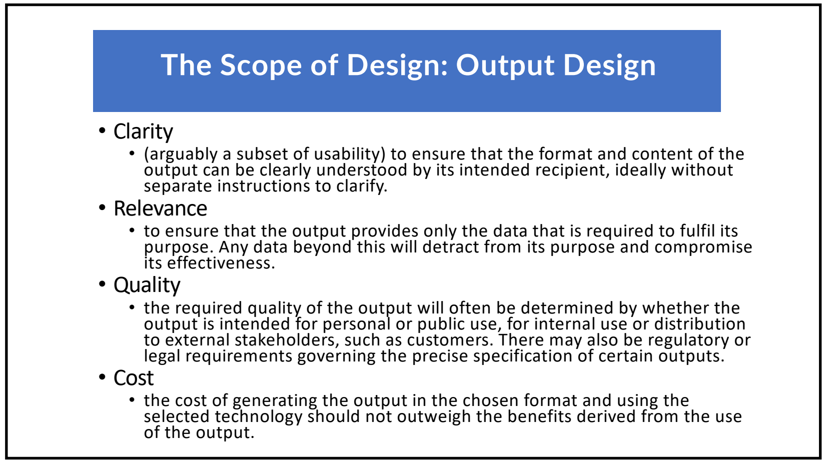
16



The Scope of Design: Output Design

- Considerations:
  - Efficiency
    - to ensure that the output makes efficient use of resources, such as mains power, processor and memory usage and consumables like paper and toner/ink.
  - Reliability and timing
    - to ensure that the outputs are available when they are needed.
  - Accuracy
    - to ensure that any data output from the system is correct and up to date.
  - Usability
    - to ensure that the outputs are easy for the user to obtain and interpret.

17



The Scope of Design: Output Design

- Clarity
  - (arguably a subset of usability) to ensure that the format and content of the output can be clearly understood by its intended recipient, ideally without separate instructions to clarify.
- Relevance
  - to ensure that the output provides only the data that is required to fulfil its purpose. Any data beyond this will detract from its purpose and compromise its effectiveness.
- Quality
  - the required quality of the output will often be determined by whether the output is intended for personal or public use, for internal use or distribution to external stakeholders, such as customers. There may also be regulatory or legal requirements governing the precise specification of certain outputs.
- Cost
  - the cost of generating the output in the chosen format and using the selected technology should not outweigh the benefits derived from the use of the output.

18

Common Output Technologies

- Monitor (CRT, LCD, LED, TFT, OLED, Plasma, UHD)
- Touch screen (including smartphones and tablets)
- Printer (laser, ink-jet, thermal, impact)
- Plotters
- Speakers and headphones
- SMS & MMS
- Email
- Digital media (magnetic, solid-state, optical)
- XML

19

Choosing Appropriate Output Technologies

- When choosing appropriate output technologies, the designer must pay particular attention to the following:
  - The application itself
  - Circumstances
    - User and environmental constraints also influence the selection of the output device. Sight-impaired people will require special consideration and audio or touch devices may be needed. The location of the information requirement may demand output devices that have certain security features, are tolerant to dirt and dust or which are mobile and can be handheld.
  - Cost
    - Budgets are usually a constraint on system development projects, so the designer must be careful to select technologies that are justified and deliver benefits that outweigh their cost.

20

Output User Interfaces

- On demand
  - usually produced to satisfy ad hoc queries about data stored in the system
- Summary
  - These only provide totals from a set of more detailed data records
- Exception
  - These reports highlight exceptional or extraordinary situations to prompt investigation or action
- Data dumps
  - Often data is output as a file for use by a software-based reporting or analysis tool
- Archive
  - Some reports are produced when the appropriate data is no longer required on the system

21

System to System Data Interchange

- The I/O design task often needs to consider how the data will be extracted from a source system and be transmitted to a destination system, and then converted into a form that can be used by the system
- The functional model for the new system should identify the system boundary and define any interfaces with other systems.
- The ability for systems to interface with each other is referred to as interoperability and, traditionally, systems exchanged data using a mechanism known as EDI (Electronic Data Interchange). However, more recent developments use standard protocols to effect a point-to-point exchange of data using XML data files and schemas, often with the aid of a specialist piece of software referred to as middleware.

22

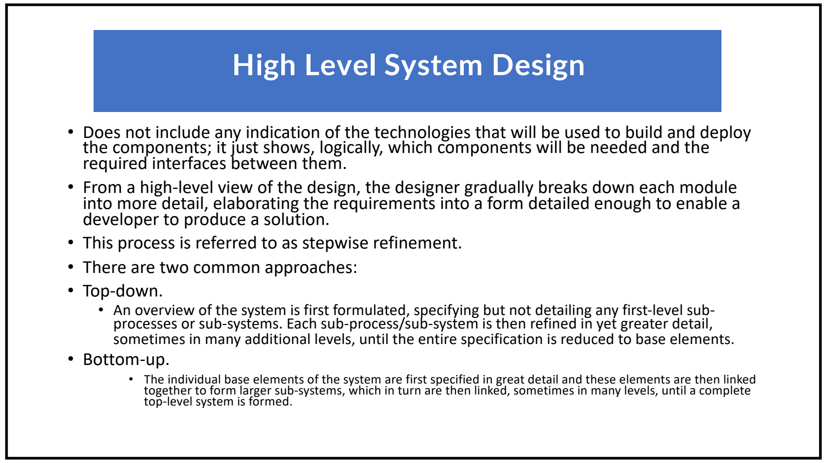
PROCESS DESIGN

23

Process Design

- The principal objective of process design is to specify how the functional requirements defined during requirements engineering will be realised using a series of individual programs or software components that will be built and will interact with each other in order to deliver the required system behaviour.
- Process design is often sub-divided into two discrete activities:
  - High-level system design.
    - This is concerned with identifying the discrete components needed to realise the functional requirements, and the interfaces necessary to enable them to communicate with each other.
  - Detailed program specification.
    - This defines how each component identified in the high-level design is to be internally built. This is also referred to as module design or component engineering.
- Furthermore, the objectives:
  - To produce reusable code and the desire for an efficient, reliable, expandable, maintainable solution.

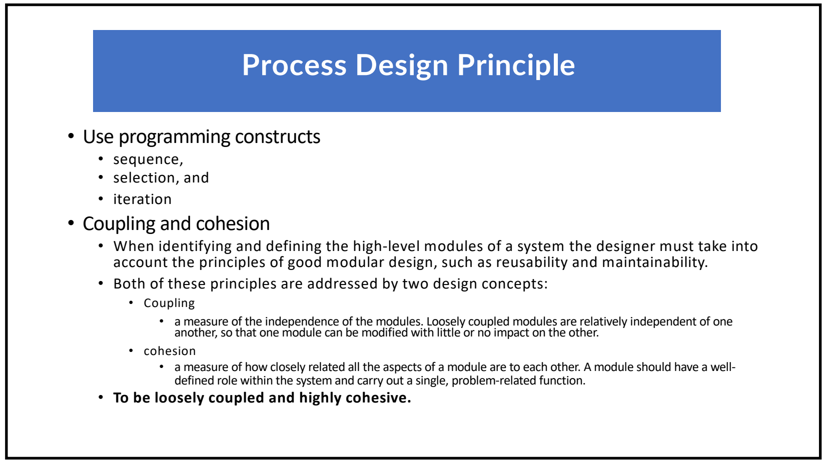
24



High Level System Design

- Does not include any indication of the technologies that will be used to build and deploy the components; it just shows, logically, which components will be needed and the required interfaces between them.
- From a high-level view of the design, the designer gradually breaks down each module into more detail, elaborating the requirements into a form detailed enough to enable a developer to produce a solution.
- This process is referred to as stepwise refinement.
- There are two common approaches:
- Top-down.
  - An overview of the system is first formulated, specifying but not detailing any first-level sub-processes or sub-systems. Each sub-process/sub-system is then refined in yet greater detail, sometimes in many additional levels, until the entire specification is reduced to base elements.
- Bottom-up.
    - The individual base elements of the system are first specified in great detail and these elements are then linked together to form larger sub-systems, which in turn are then linked, sometimes in many levels, until a complete top-level system is formed.

25



Process Design Principle

- Use programming constructs
  - sequence,
  - selection, and
  - iteration
- Coupling and cohesion
  - When identifying and defining the high-level modules of a system the designer must take into account the principles of good modular design, such as reusability and maintainability.
  - Both of these principles are addressed by two design concepts:
    - Coupling
      - a measure of the independence of the modules. Loosely coupled modules are relatively independent of one another, so that one module can be modified with little or no impact on the other.
    - cohesion
      - a measure of how closely related all the aspects of a module are to each other. A module should have a well-defined role within the system and carry out a single, problem-related function.
  - **To be loosely coupled and highly cohesive.**

26

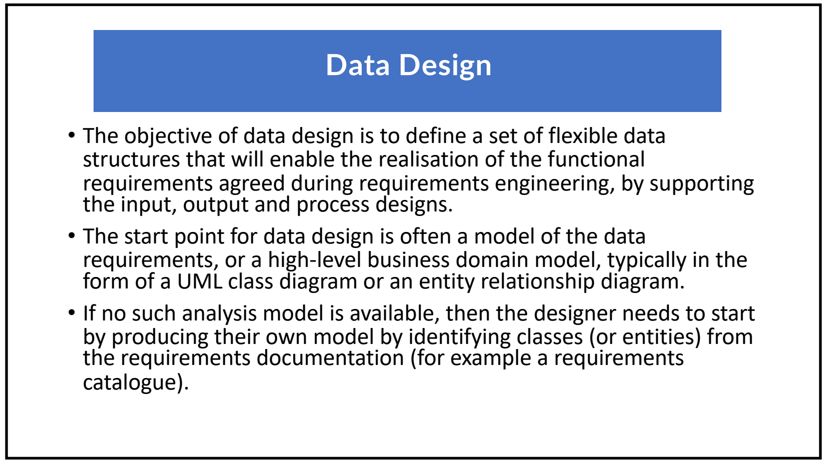
Unit/ Component Design

- The high-level design effectively treats the modules as ‘black-boxes’, where the internal workings are not visible but only the observable behaviour and any interfaces are determined.
- Once the high-level modular design is complete, the designer can turn their attention to specifying the detailed workings of each module.
- These specifications are often referred to as program specifications, as they effectively specify the work that the programmer needs to undertake.
- The most commonly used techniques for specifying these program units are flow charts (such as the UML activity diagram) and pseudocode (also called Structured English).

27

DATA DESIGN

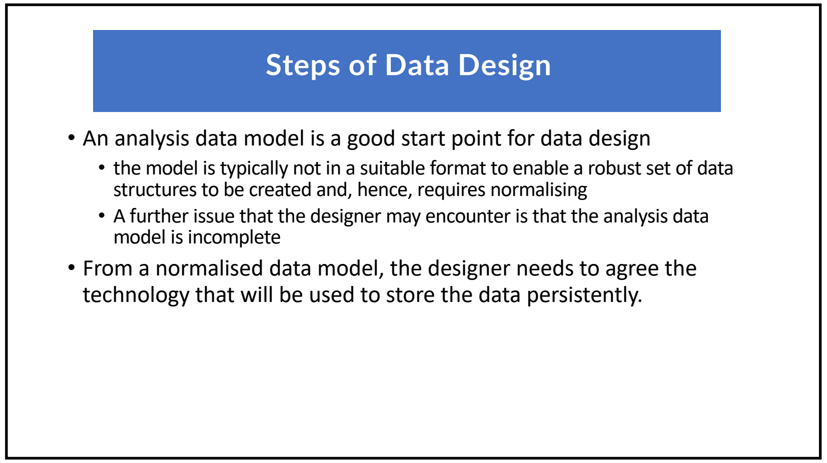
28



Data Design

- The objective of data design is to define a set of flexible data structures that will enable the realisation of the functional requirements agreed during requirements engineering, by supporting the input, output and process designs.
- The start point for data design is often a model of the data requirements, or a high-level business domain model, typically in the form of a UML class diagram or an entity relationship diagram.
- If no such analysis model is available, then the designer needs to start by producing their own model by identifying classes (or entities) from the requirements documentation (for example a requirements catalogue).

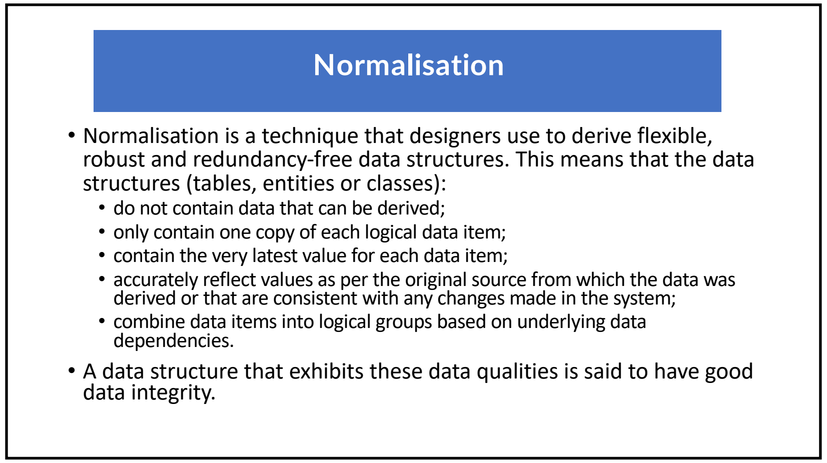
29



Steps of Data Design

- An analysis data model is a good start point for data design
  - the model is typically not in a suitable format to enable a robust set of data structures to be created and, hence, requires normalising
  - A further issue that the designer may encounter is that the analysis data model is incomplete
- From a normalised data model, the designer needs to agree the technology that will be used to store the data persistently.

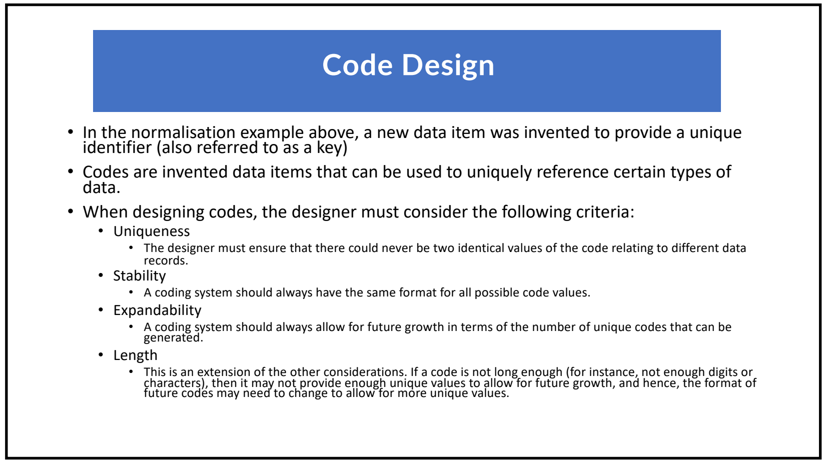
30



Normalisation

- Normalisation is a technique that designers use to derive flexible, robust and redundancy-free data structures. This means that the data structures (tables, entities or classes):
  - do not contain data that can be derived;
  - only contain one copy of each logical data item;
  - contain the very latest value for each data item;
  - accurately reflect values as per the original source from which the data was derived or that are consistent with any changes made in the system;
  - combine data items into logical groups based on underlying data dependencies.
- A data structure that exhibits these data qualities is said to have good data integrity.

31



Code Design

- In the normalisation example above, a new data item was invented to provide a unique identifier (also referred to as a key)
- Codes are invented data items that can be used to uniquely reference certain types of data.
- When designing codes, the designer must consider the following criteria:
  - Uniqueness
    - The designer must ensure that there could never be two identical values of the code relating to different data records.
  - Stability
    - A coding system should always have the same format for all possible code values.
  - Expandability
    - A coding system should always allow for future growth in terms of the number of unique codes that can be generated.
  - Length
    - This is an extension of the other considerations. If a code is not long enough (for instance, not enough digits or characters), then it may not provide enough unique values to allow for future growth, and hence, the format of future codes may need to change to allow for more unique values.

32

Types of Codes

- Simple sequential number
- Faceted codes
  - A faceted code is one that is made up of a number of parts, or facets
- Self-checking codes

33

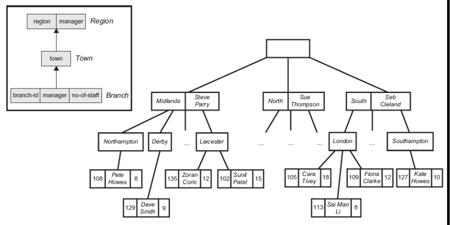
Database Management Systems

- A database is a collection of organised data sets that are created, maintained and accessed using a specialist software package called a database management system (DBMS).
- DBMSs may use any of a variety of database models, which define how the data is logically structured and, hence, how queries need to access the data. The most commonly-used models are:
  - hierarchical,
  - network,
  - relational, and
  - object-oriented

34

Hierarchical Databases

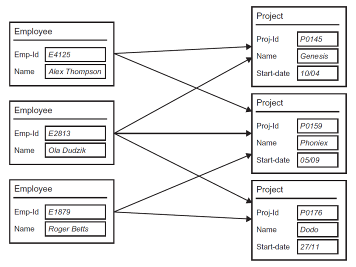
- With a hierarchical database, the data is organised as a set of records that form a hierarchy
- Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM, and now describe the structure of XML documents. Microsoft's Windows® Registry is another more recent example of a hierarchical database.



35

Network Databases

- The network database model (also known as the CODASYL database model) is a more flexible derivation of a hierarchical database. Instead of each data node being linked to just one parent node, it is possible to link a node to multiple parents, thus forming a network-like structure.
- Network databases can represent complex many-to-many relationships between the underlying data entities/classes, whereas hierarchical databases are restricted to simpler one-to-many structures
- A network database consists of a collection of records connected to each other through links.
- Probably the best known examples of a network DBMS are IDMS (Integrated Data Management System) and IDS (Integrated Data Store).



36

Relational Databases

- Hierarchical and network databases have largely been superseded by relational databases, based on the relational model, as introduced by E.F. Codd (Ted Codd) in 1970.
- This model was devised to make database management systems more independent of any particular application and, hence, more flexible for sharing databases between different applications.
- The basic data structure of the relational model is the table (also referred to as a relation), where information about a particular entity is represented in rows and columns.
- A relation is a set of rows (referred to as tuples). The columns represent the various attributes of the entity and a row is an actual instance of the entity that is represented by the relation.

Employee

Emp-Id	Name
E4125	Alex Thompson
E2813	Ola Dudzik
E1879	Roger Betts

Project

Proj-Id	Name	Start-date
P0145	Genesis	10/04
P0159	Phoenix	05/09
P0176	Dodo	27/11

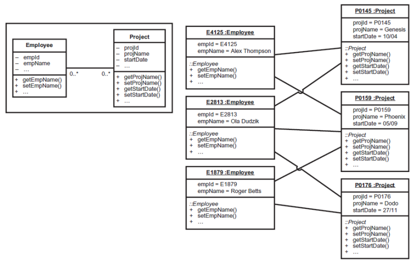
Employee-Project

*Emp-Id	*Proj-Id
E4125	P0145
E4125	P0159
E2813	P0145
E2813	P0159
E2813	P0176
E1879	P0159
E1879	P0176

37

Object Oriented Databases

- Many OO applications, whilst manipulating objects at run-time, need to hold a persistent copy of the data that is encapsulated within the objects, in some form of permanent store.
- Hence, many OO systems use relational databases to store their data persistently, a technique referred to as object-relational mapping.
- Object-oriented database management systems (OODBMS) started to evolve from research in the early to mid 1970s, with the first commercial products appearing around 1985.
- These new database technologies enabled OO developers to retain the objects that they manipulated at run-time, reducing the need for object-relational mapping.
- They introduced the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.
- OO databases store data as a series of objects that are connected to each other by pointers. This means that the actual data can be directly accessed only through the invocation of one of the object's operations, which can then build in security and integrity rules to ensure that the data cannot be corrupted or accessed by unauthorised users.



38

Physical Files

- Some systems require that they directly manipulate data files stored on disk.
- When designing data file structures, two fundamental factors must be considered:
  - 1. how to physically organise the data on a disk (disk being the most utilised of permanent storage media);
  - 2. how to access/retrieve the stored data.

39

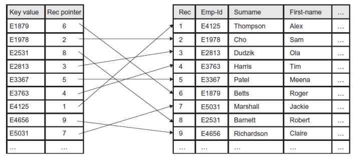
File Organisations

- Serial
  - The records contained in the file are physically organised in the order in which they are created (in other words, in no logical sequence)
  - Advantages : simple to add to, a new record needs to be added to the file it is simply appended to the end of the file, requiring no re-organisation
  - Disadvantages : would need to start retrieving the data with the first record and then keep skipping through the records until the required data had been found.
- Sequential
  - The data records are physically stored on disk in some particular sequence
  - Advantages : much easier to access/retrieve. This is because the application knows the sequence and can therefore perform some form of search algorithm to help reduce the time that would have been spent skipping through each record one at a time
  - Disadvantages : adding new records to a sequential file necessitates a complete re-organisation of the file each time, which can be extremely time-consuming.

40

File Access/Retrieval Mechanisms

- The most popular approach to manipulating data in physical files is
  - to store the actual data in a serial file,
  - whilst using a sequential file as an index into the serial file
- The concept of an index file is essentially the same as using an index in a book. To find the entry that you are looking for, you first look it up in the index to find the corresponding page number, and then you can skip quickly to the required page. The same is true when accessing data using an index file.
- The index file is a sequential file where the records are physically organised in the order of the key values.
- Consequently, the index file can be quickly scanned using an algorithm such as the binary chop, mentioned above, so that a pointer to the actual data record can be obtained.



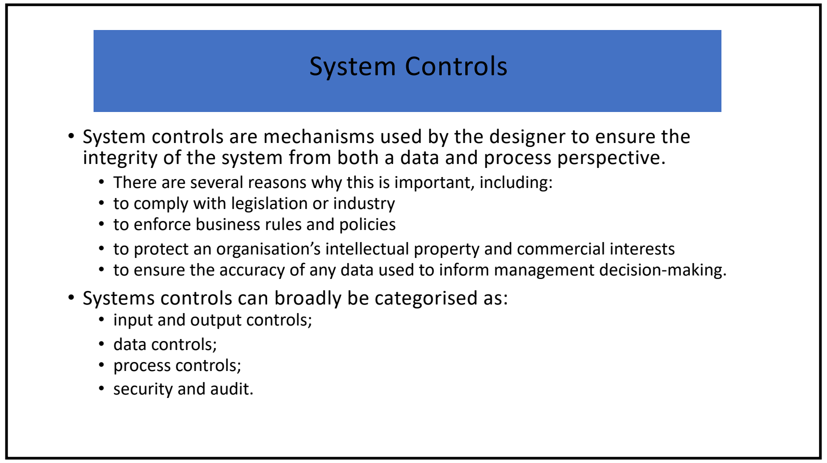
Key value	Rec pointer
E1879	6
E1978	2
E2531	8
E2813	3
E3367	5
E3763	4
E4125	1
E4656	9
E5031	7
...	...

Rec	Emp-Id	Surname	First-name	...
1	E4125	Thompson	Alex	...
2	E1978	Cho	Sam	...
3	E2813	Dudzik	Ola	...
4	E3763	Harris	Tim	...
5	E3367	Patel	Meena	...
6	E1879	Betts	Roger	...
7	E5031	Marshall	Jackie	...
8	E2531	Barnett	Robert	...
9	E4656	Richardson	Claire	...
...	...	...	...	...

41

SECURITY AND CONTROL DESIGN

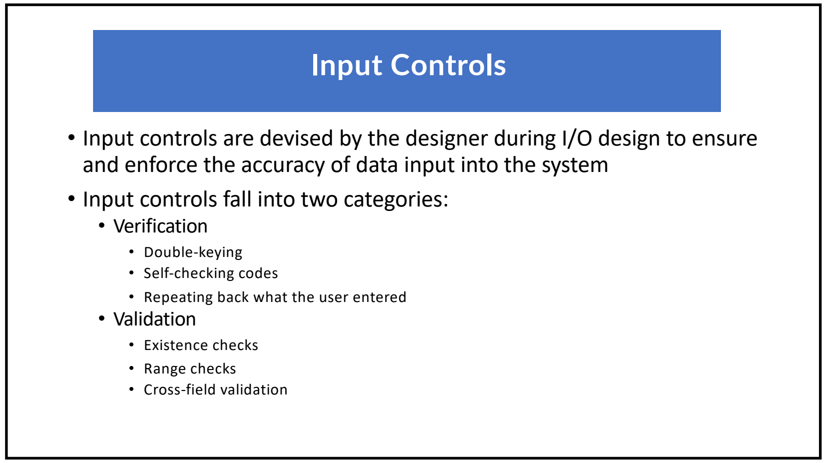
42



System Controls

- System controls are mechanisms used by the designer to ensure the integrity of the system from both a data and process perspective.
  - There are several reasons why this is important, including:
  - to comply with legislation or industry
  - to enforce business rules and policies
  - to protect an organisation’s intellectual property and commercial interests
  - to ensure the accuracy of any data used to inform management decision-making.
- Systems controls can broadly be categorised as:
  - input and output controls;
  - data controls;
  - process controls;
  - security and audit.

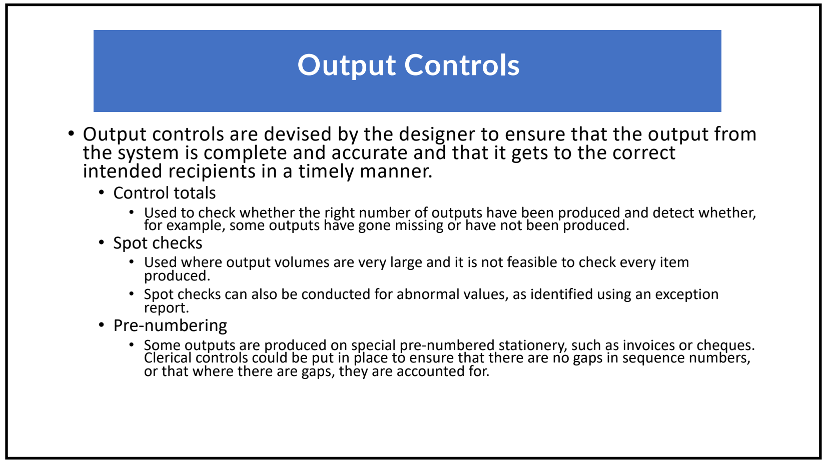
43



Input Controls

- Input controls are devised by the designer during I/O design to ensure and enforce the accuracy of data input into the system
- Input controls fall into two categories:
  - Verification
    - Double-keying
    - Self-checking codes
    - Repeating back what the user entered
  - Validation
    - Existence checks
    - Range checks
    - Cross-field validation

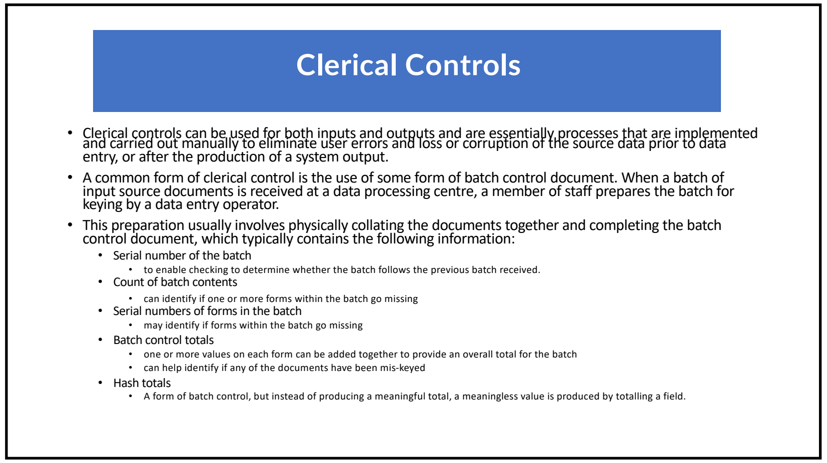
44



Output Controls

- Output controls are devised by the designer to ensure that the output from the system is complete and accurate and that it gets to the correct intended recipients in a timely manner.
  - Control totals
    - Used to check whether the right number of outputs have been produced and detect whether, for example, some outputs have gone missing or have not been produced.
  - Spot checks
    - Used where output volumes are very large and it is not feasible to check every item produced.
    - Spot checks can also be conducted for abnormal values, as identified using an exception report.
  - Pre-numbering
    - Some outputs are produced on special pre-numbered stationery, such as invoices or cheques. Clerical controls could be put in place to ensure that there are no gaps in sequence numbers, or that where there are gaps, they are accounted for.

45



Clerical Controls

- Clerical controls can be used for both inputs and outputs and are essentially processes that are implemented and carried out manually to eliminate user errors and loss or corruption of the source data prior to data entry, or after the production of a system output.
- A common form of clerical control is the use of some form of batch control document. When a batch of input source documents is received at a data processing centre, a member of staff prepares the batch for keying by a data entry operator.
- This preparation usually involves physically collating the documents together and completing the batch control document, which typically contains the following information:
  - Serial number of the batch
    - to enable checking to determine whether the batch follows the previous batch received.
  - Count of batch contents
    - can identify if one or more forms within the batch go missing
  - Serial numbers of forms in the batch
    - may identify if forms within the batch go missing
  - Batch control totals
    - one or more values on each form can be added together to provide an overall total for the batch
    - can help identify if any of the documents have been mis-keyed
  - Hash totals
    - A form of batch control, but instead of producing a meaningful total, a meaningless value is produced by totalling a field.

46

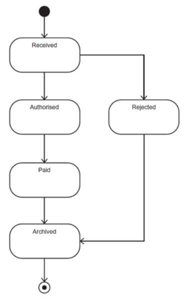
Data Control

- Data controls in terms of validation. Validation rules are typically applied at individual data item level, or across two data items, in the case of cross-field validation.
- The data dictionary would define the following information about each data item, which effectively imposes constraints on the data items that must be enforced by system controls and tested during systems testing:
  - Data type
  - Length
  - Format
  - Value range
  - Mandatory
  - Default value
  - Visibility

47

Process Control

- System controls are devised to enforce business rules, which can be defined within a data model.
- However, there are other models produced both during systems analysis and systems design.
- For example, many systems enforce some form of process workflow, which can be explicitly captured in some form of state model, such as a UML state machine diagram or a state transition diagram.



48

Security And Audit

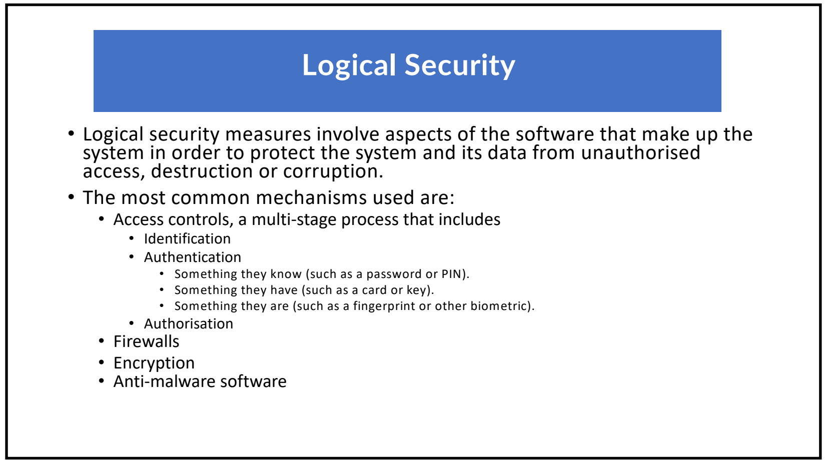
- Security is a special kind of system control that is devised to prevent unauthorised access to the system or to certain functions or data held within the system, and prevent loss of, or corruption to, system artefacts as a result of a range of risks.
- In terms of security, the options available to the designer can be broadly categorised as **physical** or **logical security**.
- However, there is a further aspect to security design that is not about the prevention of access or loss of system artefacts, but concerns the traceability of user activity: **systems audit**.

49

Physical Security

- Physical security measures prevent direct access to computer equipment, including workstations, servers and backing storage.
- The most common mechanisms used are:
  - Locks
  - Barriers
  - Safes
  - Security Personnel

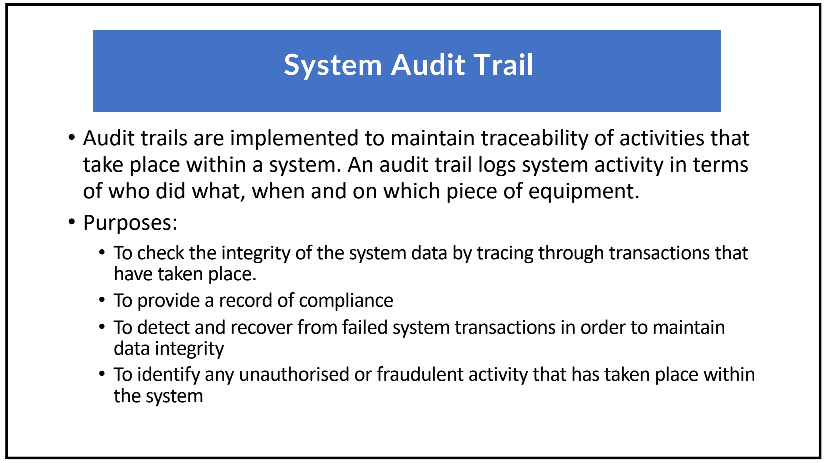
50



Logical Security

- Logical security measures involve aspects of the software that make up the system in order to protect the system and its data from unauthorised access, destruction or corruption.
- The most common mechanisms used are:
  - Access controls, a multi-stage process that includes
    - Identification
    - Authentication
      - Something they know (such as a password or PIN).
      - Something they have (such as a card or key).
      - Something they are (such as a fingerprint or other biometric).
    - Authorisation
  - Firewalls
  - Encryption
  - Anti-malware software

51



System Audit Trail

- Audit trails are implemented to maintain traceability of activities that take place within a system. An audit trail logs system activity in terms of who did what, when and on which piece of equipment.
- Purposes:
  - To check the integrity of the system data by tracing through transactions that have taken place.
  - To provide a record of compliance
  - To detect and recover from failed system transactions in order to maintain data integrity
  - To identify any unauthorised or fraudulent activity that has taken place within the system

52

Audit Records

- unique sequence number;
- date of the transaction;
- time of the transaction;
- user ID of the person making the transaction;
- machine ID of the device used to make the transaction;
- type of transaction (such as Payment, Credit Note, Invoice, and so on);
- transaction value(s);
- data value(s) before transaction;
- data value(s) after transaction.

53

LOGICAL AND PHYSICAL DESIGN

54

Logical & Physical Design

- Design activities can be divided into two separate stages of design:
  - logical design
    - ‘platform-agnostic’ insofar as it makes no specific reference to the implementation environment or technologies
  - physical design
    - defines how the logical solution elements (inputs, outputs, processes and data) are to be built and deployed using specific technologies and infrastructure of the target implementation environment
- Benefit of splitting design into these two separate stages is that a generic solution can be proposed that can be adapted for implementation in a range of different environments.
- This is how the design objective of portability is achieved.

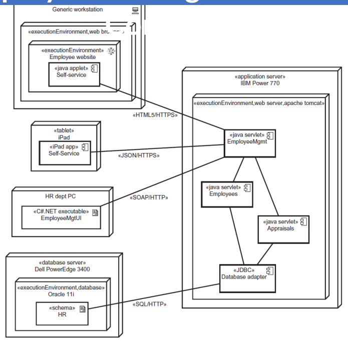
55

Physical Data Design & Physical Process Design

- Physical data design
  - involves taking the normalised data model produced during logical data design and ‘tuning’ it for implementation using specific data storage technologies and media
  - during this logical-to-physical mapping exercise, it is common to ‘de-normalise’ the logical data model in order to improve system performance.
  - de-normalisation re-introduces data redundancy that was removed during the normalisation process
- Physical process design
  - considers which technologies to use in order to build the components and how they should be deployed within the implementation environment

56

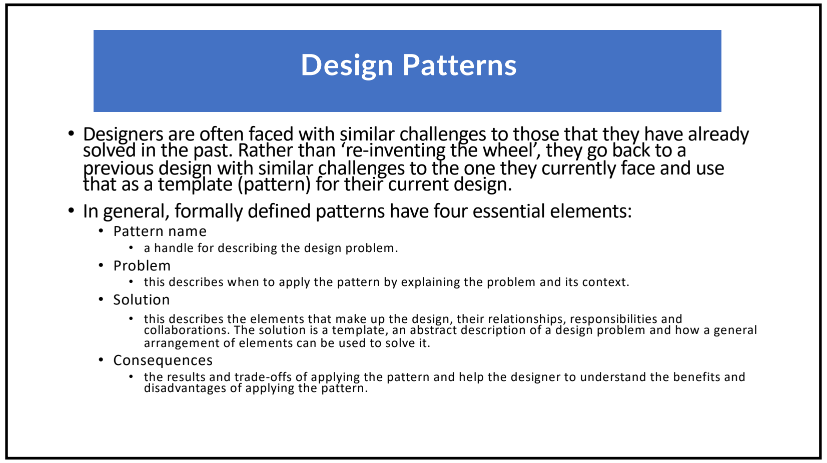
Example Of Physical Process Design
UML Deployment Diagram Showing Physical



57

DESIGN PATTERN

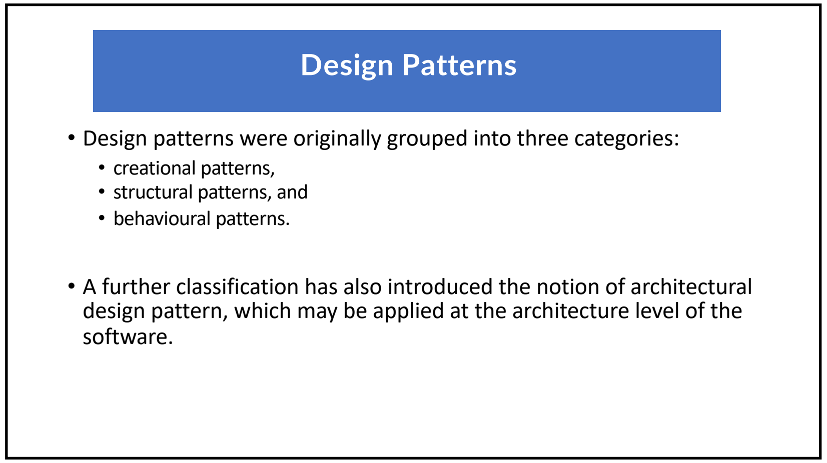
58



Design Patterns

- Designers are often faced with similar challenges to those that they have already solved in the past. Rather than 're-inventing the wheel', they go back to a previous design with similar challenges to the one they currently face and use that as a template (pattern) for their current design.
- In general, formally defined patterns have four essential elements:
  - Pattern name
    - a handle for describing the design problem.
  - Problem
    - this describes when to apply the pattern by explaining the problem and its context.
  - Solution
    - this describes the elements that make up the design, their relationships, responsibilities and collaborations. The solution is a template, an abstract description of a design problem and how a general arrangement of elements can be used to solve it.
  - Consequences
    - the results and trade-offs of applying the pattern and help the designer to understand the benefits and disadvantages of applying the pattern.

59



Design Patterns

- Design patterns were originally grouped into three categories:
  - creational patterns,
  - structural patterns, and
  - behavioural patterns.
- A further classification has also introduced the notion of architectural design pattern, which may be applied at the architecture level of the software.

60