• Describe what is meant by the term information system, what it solves. Give examples.

IS is the interaction between people, processes and data.

Is designed to process (retrieve, transmit, store, search, manipulate, display) information. (Economic, personal, payroll system, Warehauses, CRM, online store, etc.)

• Describe what is meant by an information system domain and give examples.

By the domain we understand a group of related "things, concepts and terms" in the sense of a customer or user point of view. (Bank, Insurance, Hospital)

• What questions must we answer when implementing an information system? Give examples of answers to each question in the context of a specific information system.

Key questions: • WHAT? in the sense of data. • HOW? the processes performed with the information. • WHERE? places where information is worked with. • WHO? Who works with information and in what role. • WHEN? When and on what impulses information is worked with • WHY? It is about aims and rules to achieve the requested goals.

Example:??

• From what perspectives (level of abstraction) can we look at an information system? And in what roles? • What is meant by the architecture of an information system and what does it include?

Levels of Abstraction – Strategic decision, Business model, System model, Technology, Operations. Roles: Planner, Owner, Designer, Bulder, Programmer, User

• What are the differences between static, dynamic and mobile information system architecture? Give examples.

Architecture:

Static: The structure of the system is given at design time (Simple IS)

Dynamic: Supports creation, modification and termination components and links at runtime. Structure of IS is changed dynamically. (Component based IS)

Mobile: Extends the dynamic architecture to include mobile elements. Components and link aremoved at runtime accordind to the state of computation. (Mobile based IS)

• What does the structure of each information system contain? Give examples.

Architecture of IS : the basic organization of a software system, including its components, their interrelationships and relationships with the system environment, the principles of the design of such a system and its development. Contain: app domain view, developers view, view of data and flows, physical layout of components.

• Describe what is meant by an information system component. Give examples

Components of IS - parts which interact to do functionality (HW, SW, Network, DB, Human resource).

• What is the difference between information system architecture and information system design?

• What is the correct procedure for determining the architecture and design of an information system?

How to determine architecture: Identify system requriments, divide IS into components, assign requirements to the components, Verification that requirements has been allocated. Use standards.

• Which competencies does an information system contain? Give examples of these competencies.

Core competencies: Communication with the user (presenting information) gathering requirements Information processing and (temporary) storage. Main purpose of IS Pernament store of information (data). Persistence to DB storage)

Lecture 3:

• What is meant by a design pattern? What does each pattern contain? Give examples.

Design patterns represent the best practices used by experienced object-oriented software developers. It usually contain: • Concise name (title) • Problem (What) • Context (Who, When, Where, Why) • Solution including UML diagrams (How) • Examples including source codes

Example: singleton

• Describe the essence of three-layer architecture. What is the difference between a physical and a logical three-layer architecture?

Three layer architecture is well-established software application architecture that organizes applications into three logical layers and/or physical computing tiers. The each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team.

Difference:???

• What is the essence of the MVC pattern? How does it differ from the three-layer architecture?

• MVC – model-view-controller used for: • Separation of layers at the logical level • Dependency minimization, isolated modification of individual layers • Controller - component that enables the interconnection between the views and the model so it acts as an intermediary. • It is triangular architecture.

Differ:???

• What does the domain logic design pattern group address?

Domain logic patterns – handles the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed. Transaction Script, Domain Model, Table Module, Service Layer

• Describe the essence of each domain logic pattern (Transaction script, Domain model, Table module, Service layer).

- Transaction script - business applications is represented as a set of transactions. It organize bussines logic into procedures which hadles a single request from presentation. Examples is just method like CalculateSalary. Usefull for simple Bussines logic.

- Domain Model pattern - forms a network of interconnected objects, in which each object is a separate significant entity: it can be as big as a corporation or as small as the line from the order form. It incorporate behaviour and data. Representation is Class diagram. Usefull for easy maintenance, long term systems with complex BL. One order object is one instance of class Order.
- Table Module - pattern divides the logic of the definition domain (domain) into separate classes for each table in the database and one instance of the class contains various procedures that work with data. All orders are managed by single table module. Not too complex BL, A single instance that handles the business logic for all rows in a database table or view.
- Service layer - Defines an application‘s boundary with a layer of services that establishes a set of available operations and coordinates the application‘s response in each operation

• How do the different domain logic patterns differ from each other? When, where and why to use/not to use them? Give examples.

Differ:??? When/Why/Where????

Lecture 4:

What does the group of data patterns address?

These patterns solves the basic requirement of the independence of the domain logic from the logic of data access. Does not contain domain logic methods

• Describe the nature of each pattern for working with data sources. (table data gateway, row data gateway, active record, data mapper).

- Table data gateway object that acts as a Gateway (GoF) to a database table. One instance handles all the rows in the table.
- Row data gateway object that acts as a Gateway to a single record in a data source. There is one instance per row.
- Active record object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.
- Data mapper - layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself.

• What are the differences between the different data source patterns? When, where and why to use/not to use them?

- Difference???

• What patterns for domain logic can be worked with some of the patterns for working with data sources and why? Please provide a list of examples.

• Using Data source patterns and domain logic patterns

Table data gateway • Yes – Transaction script, Table module • Yes/No – Domain model (complexity)

Row data gateway • Yes – Transaction script • No – Domain model

Active record • Yes – Transaction script • No – Domain model

Data mapper • Yes - Domain Model • No - Active Records, Table Module

Why??? Example???

Lecture 5:

• For each of the four design patterns for working with data sources, think about it and write a piece of source code that tells you which pattern it is.

• What do the object-relational behavioral patterns address together?

These patterns are designed to support and reduce problems that are inherent in data source patterns. These problems include managing of transactions and managing of in-memory objects to avoid duplication and preserve data integrity

• When should we consider using the Unit of Work pattern and why? Explain, with an example, how to use it.

• When should we consider using the Identity Map pattern and why? Explain, with an example, how to use it.

• When should we consider using the Lazy Load pattern and why? Using an example, explain how to use it.

• Think about which patterns for working with data sources you could use together with some of the behavioral patterns and why? Try to find examples.

Unit of work: - Active Records, -Table Data Gateway, +Row Data Gateway, +Data Mapper

Identity map: + Active Records, Table Data Gateway, Row Data Gateway, Data Mapper

Lazy load: + Active Records, Table Data Gateway, Row Data Gateway, Data Mapper

Why?? Example???

Lecture 6:

• In what situations would you use the Identity field pattern and why? And when not? Write a fragment of code that shows why you used this pattern.

Identity Field - Saves a database ID field in an object to maintain identity between an in-memory object and a database row.

• What is the difference between the Foreign key mapping and Associate table mapping patterns? Write two fragments of code that show that you used these patterns.

Foreign Key Mapping - Maps an association between objects to a foreign key reference between tables.

Association Table Mapping - Saves an association as a table with foreign keys to the tables that are linked by the association.

Code???

• How does the Dependent mapping pattern differ from the Data mapper pattern? When is it appropriate to use it?

???

• Write a code snippet that shows that you have used the Dependent mapping pattern. Come up with at least two examples suitable for using the Embedded value pattern. What is the essence of this pattern? Why and when is it appropriate to use it?

Example: Salary maps into Amount and Currency. Name maps into First name and Surrname Name. Datetime into Date, Time, Day of week

Embedded Value - maps an object into several fields of another object's table.

Why when???

• Come up with at least two examples suitable for using the Serialized LOB pattern. What is the essence of this pattern? Why and when is it appropriate to use it?

Example: OrderDetails – store Customer details data into LOB in Order. Save order items into LOB of Orde
Serialized LOB - Saves a graph of objects by serializing them into a single large object (LOB), which it stores in a database field.

Why when???

Lecture 7:

• Describe the nature of Single / Class / Concrete Table Inheritance patterns and in what situations they are appropriate to use.

Single Table Inheritance - Represents an inheritance hierarchy of classes as a single table that has columns for all the fields of the various classes.

Class Table Inheritance - Represents an inheritance hierarchy of classes with one table for each class.

Concrete Table Inheritance - Represents an inheritance hierarchy of classes with one table per concrete class in the hierarchy.

• Write a code fragment that shows that you have used the Single Table Inheritance pattern.

Code???

• What is the difference between the Class and Concrete Table Inheritance patterns? Write two code snippets that show that you have used these patterns.

???

• Think of and describe an example of using the Gateway pattern.

??

• Think of and describe an example of using the Mapper pattern.

???

• Devise and describe an example using the Layer Supertype pattern. Write a code fragment using inheritance, from that shows that you have used this pattern.

A type that acts as the supertype for all types in its layer. • Use Layer Supertype when you have common features from all objects in a layer.

Examples: • Abstract class for Identity Field pattern • Abstract class for common behavior of objects of the Data pattern Mapper • Domain Object superclass for all the domain objects in a Domain Model - common features

Code???

• Devise and describe an example using the Service Stub (Mock Object) pattern. Write a code fragment that shows that you have used this pattern.

Removes dependence upon problematic services during testing. • Define access to the service with a Gateway • The Gateway should not be a class but rather a Separated Interface so you can have one implementation that calls the real service and at least one that's only a Service Stub

Example: Code: ???

Lecture 8:

• Describe what is meant by the waterfall model and what are its advantages and disadvantages.

Waterfall The first and most traditional of the plan-driven process models and addresses all of the standard life cycle phases.

• Describe what is meant by iterative and incremental development. Give an example.

Iterative design is a methodology based on an iterative process of analysis, design, implementation (prototyping), testing and redefinition of the product. • The incremental model is based on the principle of incrementally building a production increments based on iterative design. • The development combines the features of the waterfall model with those of the iterative prototyping

• What is meant by UP (unified process), what are its principles, characteristics and phases? Give examples.

• What are the four basic characteristics of agile software development that distinguish it from waterfall and other robust approaches?

Individuals and interactions over processes and tools. • Working software over comprehensive documentation. • Customer collaboration over contract negotiation. • Responding to change over following a plan.

• What are the principles and practices underlying so-called extreme programming?

Extreme Programming Explained (XP) is one of several popular Agile Proces. XP is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it.

• What are the most important characteristics of SCRUM?

????

• Describe the process typical of so-called test-driven software development. Give an example.

refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design (in the form of refactoring).

Example??

Lecture 10:

• What is domain-specific language? Why and when is it good to use it? Give examples.

In general a DSL construct a shared language between the domain people and programmers.

Why: • Improving productivity for developers. • Improving communication with domain experts. • DSL is a laguage facade over a library or framework

When ????

Example: • SQL • HTML • A programming language "library" • E.g. DateTime and support for date and time tasks. • UML diagram of a State machine. • Regular expression • Configuration file, XAML

• What is the purpose of using a domain-specific language and what are the characteristics should it have?

???

• What is the difference between external and internal domain-specific language? Give examples.

External DSL - Language separate from the main language of the application it works with. (e.g. XML config files, SQL, regular expressions)

Internal DSL - A particular way of using a general-purpose language. (Ruby, Lisp)

• What problems can we encounter when designing domain-specific domain-specific language? Give examples.

Language Cacophony • Using many languages will be much more complicated than using a single one.

Cost of Building • The cost of a DSL is the cost over the cost of building the model.

Ghetto Language • Company that's built a lot of its systems on an in-house language which is not used anywhere else.

Blinkered Abstraction (Tunnel vision) • DSL It allows you to express the behavior of a domain much more easily than if you think in terms of lowerlevel constructs.