DOWNLOAD PDF CONCEPTUAL MODEL OF UML

Chapter 1 : Conceptual model (computer science) - Wikipedia

A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other. As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually.

Next Page UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. OMG is continuously making efforts to create a truly industry standard. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. Object-oriented concepts were introduced much earlier than UML. At that point of time, there were no standard methodologies to organize and consolidate the object-oriented development. It was then that UML came into picture. There are a number of goals for developing UML but the most important is to define some general purpose modeling language, which all modelers can use and it also needs to be made simple to understand and use. UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system. The system can be a software or non-software system. Thus it must be clear that UML is not a development method rather it accompanies with processes to make it a successful system. A conceptual model can be defined as a model which is made of concepts and their relationships. A conceptual model is the first step before drawing a UML diagram. It helps to understand the entities in the real world and how they interact with each other. As UML describes the real-time systems, it is very important to make a conceptual model and then proceed gradually. An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance and the classes can also be associated in different ways as per the requirement. Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML. UML is powerful enough to represent all the concepts that exist in object-oriented analysis and design. UML diagrams are representation of object-oriented concepts only. OO Analysis and Design OO can be defined as an investigation and to be more specific, it is the investigation of objects. Design means collaboration of identified objects. Thus, it is important to understand the OO analysis and design concepts. The most important purpose of OO analysis is to identify objects of a system to be designed. This analysis is also done for an existing system. Now an efficient analysis is only possible when we are able to start thinking in a way where objects can be identified. After identifying the objects, their relationships are identified and finally the design is produced. Making a design, which can be converted to executables using OO languages. There are three basic steps where the OO concepts are applied and implemented. If these objects are identified efficiently, then the next job of design is easy. The objects should be identified with responsibilities. Responsibilities are the functions performed by the object. Each and every object has some type of responsibilities to be performed. When these responsibilities are collaborated, the purpose of the system is fulfilled. The second phase is OO design. During this phase, emphasis is placed on the requirements and their fulfilment. In this stage, the objects are collaborated according to their intended association. After the association is complete, the design is also complete. The third phase is OO implementation. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects. If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects. Once the OO analysis and design is done, the next step is very easy.

DOWNLOAD PDF CONCEPTUAL MODEL OF UML

Chapter 2 : What are Conceptual, Logical and Physical Data Models? | InfoAdvisors

For nearly ten years, the Unified Modeling Language (UML) has been the industry standard for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. As the de facto standard modeling language, the UML facilitates communication and reduces confusion among project stakeholders.

Conceptual modelling can be carried out at a very early point in the design cycle so that there is a basic understanding of how users conceive tasks and this can then be brought to bear on UI design. The ability to sketch conceptual models quickly and easily can save large amounts of time in UI design and help deliver more intuitive applications. We build mental models of concepts without thinking about it; it helps us categorize things in our lives easily and simply. So for example, when we think about appointment setting our mental model is probably a diary or a calendar â€" not a piece of software. This is true even though we probably use software for setting most of our appointments. Conceptual models are designed to help us communicate the underlying intention of our application design. Rick Sandusky and Mikel D. Copyright terms and licence: Public Domain The conceptual model for any design should be constructed right at the beginning of your design cycle. Not only will it reflect the concepts that you intend to bring to life within your mobile app but it will also explore relationships that exist between the concepts. In essence, the conceptual model can inform the UI design for your team. Why Conceptual Modelling is a Good Idea It provides a high-level understanding of how your mobile application will work It allows you to try to match the way your mobile application works with the mental models of your users. This in turn should make the application more usable and intuitive. It allows you to see how well your conceptual model matches different mental models. Someone who has never used scheduling software has probably never received an automate reminder of a meeting â€" how will you help the user model that idea mentally? Even the relabelling of one simple item may make your app more complex to use. What Goes Into a Conceptual Model? Tasks â€" your conceptual model should take into account the task models that you build. It should explain tasks in the language of the user and wherever possible try to avoid introducing alien concepts. Objects â€" what are the physical objects that a user relates to? A calendar is an object. You should be looking to identify these objects and map the relationships between them. Actions and attributes should be assigned to your objects. What will your users be able to do with these objects? What do you need to set a specific appointment? What attributes will that have? You can then look at the relationships between objects and see which actions and attributes belong to a wider subset of objects and carry them across easily. Terminology Definitions â€" standard terminology across the product will make it easier for you to help your user understand what an action is all about. If you book an appointment on one page of the mobile app â€" it should not be renamed as booking a meeting elsewhere. A single conceptual model helps you streamline the language of your app early to better support a consistent user experience. Your development team can use the conceptual model and your task models to generate use cases for the system and model how the user will interact with the application itself. The UI team can use the conceptual model to determine the best approach to UI design. The conceptual model provides both teams with the understanding of: What tasks the users will be able to conduct with the mobile app The objects that they will work with within the app The relationships that those objects have to each other The actions that users can take using the objects The attributes associated with each object The language used to describe objects, attributes, etc. They are an extension of task modelling and should be done during the early phase of design to bring the biggest value to the project. Susan Wenschenk reflects on mental and conceptual models here - https: Topics in this article:

Chapter 3 : UML 2 Class Diagrams: An Agile Introduction

UML class diagrams as a conceptual models. A conceptual model captures the important concepts and relationships in some domain. Concepts are represented by classes, while relationships are represented by associations.

Overview[edit] Models of concepts and models that are conceptual[edit] The term conceptual model is normal. It could mean "a model of concept" or it could mean "a model that is conceptual. With the exception of iconic models, such as a scale model of Winchester Cathedral , most models are concepts. But they are, mostly, intended to be models of real world states of affairs. The value of a model is usually directly proportional to how well it corresponds to a past, present, future, actual or potential state of affairs. A model of a concept is quite different because in order to be a good model it need not have this real world correspondence. Type and scope of conceptual models[edit] Conceptual models models that are conceptual range in type from the more concrete, such as the mental image of a familiar physical object, to the formal generality and abstractness of mathematical models which do not appear to the mind as an image. Conceptual models also range in terms of the scope of the subject matter that they are taken to represent. A model may, for instance, represent a single thing e. The variety and scope of conceptual models is due to then variety of purposes had by the people using them. Conceptual modeling is the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication. Also, a conceptual model must be developed in such a way as to provide an easily understood system interpretation for the models users. A conceptual model, when implemented properly, should satisfy four fundamental objectives. Figure 1 [5] below, depicts the role of the conceptual model in a typical system development scheme. It is clear that if the conceptual model is not fully developed, the execution of fundamental system properties may not be implemented properly, giving way to future problems or system shortfalls. These failures do occur in the industry and have been linked to; lack of user input, incomplete or unclear requirements, and changing requirements. Those weak links in the system design and development process can be traced to improper execution of the fundamental objectives of conceptual modeling. Conceptual model computer science As systems have become increasingly complex, the role of conceptual modelling has dramatically expanded. With that expanded presence, the effectiveness of conceptual modeling at capturing the fundamentals of a system is being realized. Building on that realization, numerous conceptual modeling techniques have been created. These techniques can be applied across multiple disciplines to increase the users understanding of the system to be modeled. Some commonly used conceptual modeling techniques and methods include: Data flow modeling[edit] Data flow modeling DFM is a basic conceptual modeling technique that graphically represents elements of a system. DFM is a fairly simple technique, however, like many conceptual modeling techniques, it is possible to construct higher and lower level representative diagrams. The data flow diagram usually does not convey complex system details such as parallel development considerations or timing information, but rather works to bring the major system functions into context. Data flow modeling is a central technique used in systems development that utilizes the structured systems analysis and design method SSADM. Entity relationship modeling Ontology oriented [edit] Entity-relationship modeling ERM is a conceptual modeling technique used primarily for software system representation. Entity-relationship diagrams, which are a product of executing the ERM technique, are normally used to represent database models and information systems. The main components of the diagram are the entities and relationships. The entities can represent independent functions, objects, or events. The relationships are responsible for relating the entities to one another. To form a system process, the relationships are combined with the entities and any attributes needed to further describe the process. These conventions are just different ways of viewing and organizing the data to represent different system aspects. Event-driven process chain[edit] The event-driven process chain EPC is a conceptual modeling technique which is mainly used to systematically improve business process flows. More specifically, the EPC is made up of events which define what state a process is in or the rules by which it operates. Depending on the process flow, the function has the ability to transform event states or link to other event driven process chains. Other

elements exist within an EPC, all of which work together to define how and by what rules the system operates. The EPC technique can be applied to business practices such as resource planning, process improvement, and logistics. Joint application development[edit] The dynamic systems development method uses a specific process called JEFFF to conceptually model a systems life cycle. JEFFF is intended to focus more on the higher level development planning that precedes a projects initialization. The JAD process calls for a series of workshops in which the participants work to identify, define, and generally map a successful project from conception to completion. This method has been found to not work well for large scale applications, however smaller applications usually report some net gain in efficiency. The petri net, because of its nondeterministic execution properties and well defined mathematical theory, is a useful technique for modeling concurrent system behavior , i. State transition modeling[edit] State transition modeling makes use of state transition diagrams to describe system behavior. These state transition diagrams use distinct states to define system behavior and changes. Most current modeling tools contain some kind of ability to represent state transition modeling. The use of state transition models can be most easily recognized as logic state diagrams and directed graphs for finite-state machines. Technique evaluation and selection[edit] Because the conceptual modeling method can sometimes be purposefully vague to account for a broad area of use, the actual application of concept modeling can become difficult. To alleviate this issue, and shed some light on what to consider when selecting an appropriate conceptual modeling technique, the framework proposed by Gemino and Wand will be discussed in the following text. However, before evaluating the effectiveness of a conceptual modeling technique for a particular application, an important concept must be understood; Comparing conceptual models by way of specifically focusing on their graphical or top level representations is shortsighted. Gemino and Wand make a good point when arguing that the emphasis should be placed on a conceptual modeling language when choosing an appropriate technique. In general, a conceptual model is developed using some form of conceptual modeling technique. That technique will utilize a conceptual modeling language that determines the rules for how the model is arrived at. Understanding the capabilities of the specific language used is inherent to properly evaluating a conceptual modeling technique, as the language reflects the techniques descriptive ability. Also, the conceptual modeling language will directly influence the depth at which the system is capable of being represented, whether it be complex or simple. The presentation method for selection purposes would focus on the techniques ability to represent the model at the intended level of depth and detail. The characteristics of the models users or participants is an important aspect to consider. The conceptual model language task will further allow an appropriate technique to be chosen. The difference between creating a system conceptual model to convey system functionality and creating a system conceptual model to interpret that functionality could involve to completely different types of conceptual modeling languages. Considering affected variables[edit] Gemino and Wand go on to expand the affected variable content of their proposed framework by considering the focus of observation and the criterion for comparison. The criterion for comparison would weigh the ability of the conceptual modeling technique to be efficient or effective. A conceptual modeling technique that allows for development of a system model which takes all system variables into account at a high level may make the process of understanding the system functionality more efficient, but the technique lacks the necessary information to explain the internal processes, rendering the model less effective. When deciding which conceptual technique to use, the recommendations of Gemino and Wand can be applied in order to properly evaluate the scope of the conceptual model in question. Understanding the conceptual models scope will lead to a more informed selection of a technique that properly addresses that particular model. In summary, when deciding between modeling techniques, answering the following questions would allow one to address some important conceptual modeling considerations. What content will the conceptual model represent? How will the conceptual model be presented? Who will be using or participating in the conceptual model? How will the conceptual model describe the system? What is the conceptual models focus of observation? Will the conceptual model be efficient or effective in describing the system? Another function of the simulation conceptual model is to provide a rational and factual basis for assessment of simulation application appropriateness. Models in philosophy and science[edit].

Chapter 4 : OOAD UML Analysis Model

CONCEPTUAL MODEL OF THE UML: To understand the UML, we need to form a conceptual model of the language and this requires learning three major elements.

This is depicted using a filled triangle, called a direction indicator, an example of which is shown on the offering of association between the Seminar and Course classes of Figure 5. My advice, however, is if your label is not clear, then you should consider rewording it. The arrowheads on the end of the line indicate the directionality of the association. A line with one arrowhead is uni-directional whereas a line with either zero or two arrowheads is bidirectional. Officially you should include both arrowheads for bi-directional assocations, however, common practice is to drop them as you can see, I prefer to drop them. At each end of the association, the role, the context an object takes within the association, may also be indicated. Inheritance is that mechanism. The UML modeling notation for inheritance is a line with a closed arrowhead pointing from the subclass to the superclass. Many similarities occur between the Student and Professor classes of Figure 2. Not only do they have similar attributes, but they also have similar methods. To take advantage of these similarities, I created a new class called Person and had both Student and Professor inherit from it, as you see in Figure 7. This structure would be called the Person inheritance hierarchy because Person is its root class. The Person class is abstract: Abstract classes are modeled with their names in italics, as opposed to concrete classes, classes from which objects are instantiated, whose names are in normal text. Both classes had a name, e-mail address, and phone number, so these attributes were moved into Person. The Purchase Parking Pass method is also common between the two classes, something we discovered after Figure 2 was drawn, so that was also moved into the parent class. By introducing this inheritance relationship to the model, I reduced the amount of work to be performed. Instead of implementing these responsibilities twice, they are implemented once, in the Person class, and reused by Student and Professor. For example, an airplane is made up of a fuselage, wings, engines, landing gear, flaps, and so on. Figure 8 presents an example using composition, modeling the fact that a building is composed of one or more rooms, and then, in turn, that a room may be composed of several subrooms you can have recursive composition. In UML 2, aggregation would be shown with an open diamond. Another good indication that composition makes sense is when the lifecycle of the part is managed by the whole -- for example a plane manages the activities of an engine. When deciding whether to use composition over association, Craig Larman says it best: If in doubt, leave it out. Unfortunately many modelers will agonize over when to use composition when the reality is little difference exists among association and composition at the coding level. A vocabulary defines the semantics of entity types and their responsibilities, the taxonomical relationships between entity types, and the ontological relationships between entity types. Taxonomies are classifications of entity types into hierarchies, an example of which is presented for persons Figure 9. Ontology goes beyond taxonomy. Where taxonomy addresses classification hierarchies ontology will represent and communicate knowledge about a topic as well as a set of relationships and properties that hold for the entities included within that topic. A taxonomy for people within the university. The semantics of your conceptual model are best captured in a glossary. There are several interesting aspects of Figure 9: It uses UML 2. There are three generalization sets for Person: Nationality, Role, and Gender. These generalization sets overlap - a person can be classified via each of these roles e. This is called multiple classification. Some generalization sets are mutually exclusive from others, not shown in the example, where an entity type may only be in one set. This is referred to as single classification and would be modeled using an XOR exclusive OR constraint between the two or more discriminators.

Chapter 5 : The Unified Modeling Language User Guide

Conceptual Model of the UML To understand the UML, we need to form a conceptual model of the language, and this requires learning three major elements: 1) UML's basic building blocks, 2) The rules that tell us how these building blocks may be put together and 3) Some common mechanisms that apply throughout the UML.

Things Relationships Diagrams Things are the abstractions that are first-class citizens in a model; relationships tie these things together and diagrams group interesting collections of things. Structural things Grouping things Annotational things These things are the basic object-oriented building blocks of the UML. We use them to write well-formed models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. In all, there are seven kinds of structural things: First, a class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically a class is rendered as a rectangle, usually including its name, attributes and operations as shown below: Second, an interface is a collection of operations that specify a service of a class or component. An interface therefore describes the externally visible behavior of that element. Graphically, an interface is rendered as a circle together with its name. An interface is typically attached to the class or component that realizes that interface. Therefore, collaborations have structural as well as behavioral dimensions. Graphically, a collaboration is rendered as an ellipse with dashed lines, usually including only its name as shown below: Fourth, a use case is a description of set of sequence of actions that a system performs that yields an observable result of value to a particular actor. A use case is used to structure the behavioral things in a model. A use case is realized by a collaboration. Graphically, a use case is rendered as an ellipse with solid lines, usually including only its name as shown below: Fifth, an active class is a class whose object own one or more processes or threads and therefore can initiate control activity. An active class is just like a class except its objects represent elements whose behavior is concurrent with other elements. Graphically, an active class is rendered just like a class, but with thick lines, usually including its name, attributes and operations as shown below: Sixth,a component is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Generally, a component is rendered as a rectangle with tabs, usually including its name as shown below: Seventh, a node is a physical element that exists at run time and represents a computational resource, generally having at least some memory and, often, processing capability. Graphically, a node is rendered as a cube, usually including only its name as shown below: These are the verbs of a model, representing behavior over time and space. In all, there are two primary kinds of behavioral things. First, an interaction is a behavior contains a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves messages, action sequences and links. Graphically, a message is rendered as a directed line, almost always including the name of its operation as shown below: Second, a state machine is a behavior that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events. A state machine involves other elements like states, transitions, events and activities. Graphically, a state is rendered as a rounded rectangle, usually including its name and its substates as shown below: These are the boxes into which a model can be decomposed. In all, there is one primary kind of grouping thing, namely, packages. A package is general purpose mechanism for organizing elements into groups. Structural things, behavioral things and even other things may be placed in a package. Graphically, a package is rendered as a tabbed folder, usually including only its name and sometimes, its contents as shown below: These are the comments we may apply to describe, illuminate and remark about any element in a model. There is one primary kind of annotational thing, called a note. Graphically a note is rendered as a rectangle with a dog-eared corner, together with a textual or graphical comment as shown below: Realization These relationships are the basic relational building blocks of the UML. First, a dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing. Graphically, a dependency is rendered as a dashed line, possibly directed and occasionally including a label as shown below: Second, an association is a structural relationship that describes a set of links, a link being a

connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts. Graphically, an association is rendered as a solid line, possibly directed, occasionally including a label, and often containing other adornments such as multiplicity and role names as shown below: Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent as shown below: Fourth, a realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. Graphically, a realization relationship is rendered as a cross between a generalization and a dependency relationship as shown below: Diagrams in UML A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of things and relationships. We draw diagrams to visualize a system from different perspectives. So, a diagram is a projection into a system. In theory, a diagram may contain any combination of things and relationships. In practice, however a small number of common combinations arise, which are consistent with the five most useful views that make up the architecture of a software-intensive system. For this reason, UML includes nine such diagrams:

Chapter 6 : Conceptual Data Model [Enterprise Architect User Guide]

The process of using a conceptual model can be compared to the nursing processes which consist to assessing the needs of a patient, diagnosing needs, planning needs of the patient, implementation of the plan, and evaluating the success of the plan.

Once you have grasped these ideas, you will be able to read UML models and create some basic ones. As you gain more experience in applying the UML, you can build on this conceptual model, using more advanced features of the language. Conceptual Model of UML: Interface - a collection of operations that specify the services rendered by a class or component. Collaboration - a collection of UML building blocks classes, interfaces, relationships that work together to provide some functionality within the system. Active Class - a class whose instance is an active object; an active object is an object that owns a process or thread units of execution Notation: C Component - a physical part typically manifests itself as a piece of software of the system. Node - a physical element that exists at run-time and represents a computational resource typically, hardware resources. Interaction - some behaviour constituted by messages exchanged among objects; the exchange of messages is with a view to achieving some purpose. Parse Behavioral Things contd. Grouping Things The organizational part of the UML model; provides a higher level of abstraction granularity. Package - a general-purpose element that comprises UML elements - structural, behavioral or even grouping things. Packages are conceptual groupings of the system and need not necessarily be implemented as cohesive software modules. Accounts Department Annotational Things: Parses user-query and builds expression stack or invokes ErrorHandler Notation: Relationships Articulates the meaning of the links between things. Dependency - a semantic relationship where a change in one thing the independent thing causes a change in the semantics of the other thing the dependent thing. Realization - a semantic relationship between two things wherein one specifies the behaviour to be carried out, and the other carries out the behaviour. Diagrams The graphical presentation of the model. Represented as a connected graph - vertices things connected by arcs relationships. UML includes nine diagrams - each capturing a different dimension of a software-system architecture. Class Diagram - the most common diagram found in OOAD, shows a set of classes, interfaces, collaborations and their relationships. Models the static view of the system. Object Diagram - a snapshot of a class diagram; models the instances of things contained in a class diagram. Models WHAT the system is expected to do. Sequence Diagram - models the flow of control by time-ordering; depicts the interaction between various objects by of messages passed, with a temporal dimension to it. Collaboration Diagram - models the interaction between objects, without the temporal dimension; merely depicts the messages passed between objects. Statechart Diagram - shows the different state machines and the events that leads to each of these state machines. Statechart diagrams show the flow of control from state to state. Component Diagram - shows the physical packaging of software in terms of components and the dependencies between them. Deployment Diagram - shows the configuration of the processing nodes at run-time and the components that live on them. Rules Specify what a well-formed model should look like. Stereotypes Used to create new building blocks from existing blocks. New building blocks are domain-specific. Tagged Values Used to add to the information of the element not of its instances. Stereotypes help create new building blocks; tagged values help create new attributes. Commonly used to specify information relevant to code generation, configuration management, etc. Constraints Used to create rules for the model. Rules that impact the semantics of the model, and specify conditions that must be met. Can apply to any element in the model - attributes of a class, relationship, etc. Summary Modeling captures the system in its entirety, along with the different dimensions of its complexity. Facilitates quick and efficient analysis and design and helps communicate the overall system architecture unambiguously. Principles of modeling lay down that: UML is a modeling language for visualising , specifying , constructing and documenting the artifacts of a software system. It is a modeling language and not a methodology or a process. Summary Structural things describe the static part of the model and are of seven types: Behavioral things describe the dynamic part of the model and are of two types: Dependency , Association , Generalisation and Realization. Diagrams are of nine types: Summary Certain

common mechanisms apply uniformly across the model. There are four such mechanisms: Notes are the most common adornments used, that add to the meaning of a classifier. Extensibility mechanisms include Stereotypes, Tagged values and constraints.

Chapter 7 : Using UML Diagrams for a Game Project | Studytonight

77 videos Play all UML Tutorials Point (India) Pvt. Ltd. 5 GRAMMAR MISTAKES you didn't know you were making - Duration: LetThemTalkTV Recommended for you.

Like all good data architects, I want to define the terms I use on this blog, speaking engagements, and on my projects. My friend Graeme Simsion has even done research in this naming conflict. My uses of conceptual, logical, and physical come from the Information Engineering IE methods of data modeling. Other uses and definitions arise from the database schema and academic world. The industry as a whole tends to use the IE definitions, so I tend to stick to them because they are used by the vast majority of practitioner data modelers and other team members. Conceptual Data Model A conceptual data model is a summary-level data model that is most often used on strategic data projects. It typically describes an entire enterprise. Due to its highly abstract nature, it may be referred to as a conceptual model. Common characteristics of a conceptual data model: Enterprise-wide coverage of the business concepts. Designed and developed primarily for a business audience Contains around entities or concepts with no or extremely limited number of attributes described. Sometimes architects try to limit it to printing on one page. Contains relationships between entities, but may or may not include cardinality and nullability. Entities will have definitions. Designed and developed to be independent of DBMS, data storage locations or technologies. In fact, it would address digital and non-digital concepts. This means it would model paper records and artifacts as well as database artifacts. Logical Data Model A logical data model is a fully-attributed data model that is independent of DBMS, technology, data storage or organizational constraints. It typically describes data requirements from the business point of view. While common data modeling techniques use a relational model notation, there is no requirement that resulting data implementations must be created using relational technologies. Common characteristics of a logical data model: Typically describes data requirements for a single project or major subject area. May be integrated with other logical data models via a repository of shared entities Typically contains entities, although these numbers are highly variable depending on the scope of the data model. Contains relationships between entities that address cardinality and nullability optionality of the relationships. In fact, it may address digital and non-digital concepts. Data attributes will typically have datatypes with precisions and lengths assigned. Data attributes will have nullability optionality assigned. Entities and attributes will have definitions. In fact, the diagram of a logical data model may show only a tiny percentage of the meta data contained within the model. A logical data model will normally be derived from and or linked back to objects in a conceptual data model. Physical Data Model A physical data model is a fully-attributed data model that is dependent upon a specific version of a data persistence technology. Common characteristics of a physical data model: Typically describes data requirements for a single project or application. Sometimes even a portion of an application. May be integrated with other physical data models via a repository of shared entities Typically contains tables, although these numbers are highly variable depending on the scope of the data model. Contains relationships between tables that address cardinality and nullability optionality of the relationships. Designed and developed to be dependent on a specific version of a DBMS, data storage location or technology. Columns will have nullability optionality assigned. Tables and columns will have definitions. Will also include other physical objects such as views, primary key constraints, foreign key constraints, indexes, security roles, store procedures, XML extensions, file stores, etc. The diagram of a physical data model may show only a tiny percentage of the meta data contained within the model.

DOWNLOAD PDF CONCEPTUAL MODEL OF UML

Chapter 8 : CONCEPTUAL MODEL of UML |authorSTREAM

In UML notation, the conceptual model is often described with a class diagram in which classes represent concepts, associations represent relationships between concepts and role types of an association represent role types taken by instances of the modelled concepts in various situations.

UML class diagrams as a conceptual models A conceptual model captures the important concepts and relationships in some domain. Concepts are represented by classes, while relationships are represented by associations. Packages represent groups of related classes and associations. Objects represent specific instances of classes. For example a business might organize its concepts and relationships into three packages: Classes In the hr package we might find the Position and Person classes. Typical attributes of persons and positions can be listed in the attribute compartments of the respective class icons. Typical operations performed by persons and positions can be listed in their operation compartments: In this example, an object representing a position, vice president for example, has a title attribute of type String and an execute operation that takes a task, t, as an input, and outputs a result. An object representing a person has four attributes: The person class has no operations specified at this point. It is often the case that operations correspond to responsibilities and are not assigned to classes until the design phase. For this reason operation compartments are often suppressed in domain models. Attribute compartments can also be suppressed if they are not needed. Note that the "-" sign indicates that the scope of an attribute or association endpoint is private, visible only inside of the class. Other possible scopes are: Associations The relationship "Position X is filled by person Y" can be represented by an association connecting Position and Person: Note that the name of an association can be decorated by a tiny arrow showing which way the association is read. This helps distinguish a relationship from its inverse. Note that the endpoints of an association can be decorated by: For example, an employee refers to his position as his "role" while a position refers to the employees filling it as its "actors. Objects Specific persons and positions can be represented using objects. Specific instances of the filledBy relationship can be represented by links connecting the participating objects: Abox versus Tbox UML is only one of many languages for modeling domains. For example, we can simply write down all of the facts about a domain using formulas. This is essentially what Euclid did for the domain of geometry. A collection of formulas describing a domain is called a knowledge base or ontology. Ontologies are often divided into two parts: The Tbox is the terminology component of a domain model. It defines the vocabulary of the domain. The Abox is the assertion component of a domain model. It collects together specific facts about the domain. For example, compare the statements: A student is a person who is currently enrolled in a school. Ted Williams is a student at Boston.

Chapter 9 : What is Data Modelling? Conceptual, Logical, & Physical Data Models

Structural Things â^' These are the nouns of the UML models representing the static elements that may be either physical or conceptual. The structural things are class, interface, collaboration, use case, active class, components, and nodes.

Why use Data Model? The primary goal of using data model are: Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results. A data model helps design the database at the conceptual, physical and logical levels. Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures. It provides a clear picture of the base data and can be used by database developers to create a physical database. It is also helpful to identify missing and redundant data. Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster. Types of Data Models There are mainly three different types of data models: This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures. This model is typically created by DBA and developers. The purpose is actual implementation of the database. Conceptual Model The main aim of this model is to establish the entities, their attributes, and their relationships. In this Data modeling level, there is hardly any detail available of the actual Database structure. The 3 basic tenants of Data Model are Entity: A real-world thing Attribute: Characteristics or properties of an entity Relationship: Dependency or association between two entities For example: Customer and Product are two entities. Customer number and name are attributes of the Customer entity Product name and price are attributes of product entity Sale is the relationship between the customer and product Characteristics of a conceptual data model Offers Organisation-wide coverage of the business concepts. This type of Data Models are designed and developed for a business audience. The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world. Logical Data Model Logical data models add further information to the conceptual model elements. It defines the structure of the data elements and set the relationships between them. The advantage of the Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic. At this Data Modeling level, no primary or secondary key is defined. At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships. Characteristics of a Logical data model Describes data needs for a single project but could integrate with other logical data models based on the scope of the project. Designed and developed independently from the DBMS. Data attributes will have datatypes with exact precisions and length. Normalization processes to the model is applied typically till 3NF. It offers an abstraction of the database and helps generate schema. This is because of the richness of meta-data offered by a Physical Data Model. This type of Data model also helps to visualize database structure. Characteristics of a physical data model: The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope. Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships. Developed for a specific version of a DBMS, location, data storage or technology to be used in the project. Columns should have exact datatypes, lengths assigned and default values. Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. Advantages and Disadvantages of Data Model: Advantages of Data model: The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately. The data model should be detailed enough to be used for building the physical database. The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures. Data Model helps business to communicate the within and across organizations. Data model helps to documents data mappings in ETL process Help to recognize correct sources of data to populate the model

Disadvantages of Data model: To developer Data model one should know physical data stored characteristics. This is a navigational system produces complex application development, management. Thus, it requires a knowledge of the biographical truth. Even smaller change made in structure require modification in the entire application. There is no set data manipulation language in DBMS. Conclusion Data modeling is the process of developing data model for the data to be stored in a Database. Data Models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data. There are three types of conceptual, logical, and physical. The main aim of conceptual model is to establish the entities, their attributes, and their relationships. Logical data model defines the structure of the data elements and set the relationships between them. A Physical Data Model describes the database specific implementation of the data model. The biggest drawback is that even smaller change made in structure require modification in the entire application.