**Glossary of Terms**

**Module 2 Course 4: Process Modeling**

**Activity**

Tasks or actions of work that are not very extensive in scope, such as specific day to day procedures or tactical work performed by a limited number of people. Higher level activities can be broken down into specific tasks or steps to get the work done.

**Activity Diagram**
A UML diagram that shows the sequence of activities within a process. Where multiple players are involved the diagram may be divided into swim lanes. See example below: 

Source: <http://www.agilemodeling.com/style/activityDiagram.htm>

**Actors**

People, or at times other systems, that interact with a specified system to accomplish some goal. Actors initiate the use cases of the system (a function or major activity). A use case within the system may have more than one actor, since more than one type of role may initiate the functionality of the system.

**Aggregation**

A type of relationship within UML models where one object is part of another, such as students aggregate to make up a course and several courses aggregate to make up a curriculum. This type of relationship, whole-part, tends to be somewhat weak since each object can still be somewhat independent of one another.

**Artifacts**

Work products, documents, reports, additional references or groupings that are noted in the context of a BPMN model. Artifacts may be shown as annotations within the model and they are not the core focus of what should be conveyed in the model. Two examples of artifacts are noted below:



**Balancing Feedback Loop**

A type of loop within a System Dynamics model, attempting to bring equilibrium within the system. This occurs because the sinks or states that receive flow are out of balance. Balancing feedback loops can be either positive or negative. They are extremely important to complex systems such as the earth or eco-systems.

**Business Process Modeling Notation**

A graphical standard used to model processes, usually framed in the context of a Swim Lane Diagram. BPMN is promulgated by the Object Management Group. The standards include specific ways of expressing messages that often start and stop a process or ways of showing how workflows are controlled through gateways.

**Cardinality**

Similar to Multiplicities, an indication of occurrences that takes place between objects. Usually involves data within a data model or database environment. Three types of cardinality are often defined for databases:

1:1 - For each occurrence of data X, you have one occurrence of data Y.

1:M – For each occurrence of data X, you have many occurrences of data Y.

M:M – For many occurrences of data X, you have many occurrences of data Y.

**Casual Loop Diagram**

A diagram that shows the structure of a system using casual loops since most relationships are non-linear. Large scale systems have a lot of dynamics and changes and thus, cause-effect relationships are shown in a casual way; i.e. loops.

**Class Diagram**
A UML diagram that describes the structure of a system, broken down into classes. Each class is further broken down into its attributes and operations. All classes are connected through relationships with additional notational information added. Class Diagrams are sometimes referred to as Structural Diagrams since they give a programmer the structure of how an application should be programmed.



**Component Diagram**

A system related diagram built using the Unified Modeling Language (UML), showing the relationship of major system components such as servers, middleware, and interfaces within the system. Component diagrams give you insights into what is “executable” within the system, such as the operating system and software applications involved. Component diagrams are very descriptive of a system’s structure and help developers understand how the system will have to get implemented.

**Constraint**

A restriction or condition on how an object behaves within the model and / or with other objects in the model. For example, you may constrain the variable phone number to be all numeric or you may constrain the relationship between objects, such as one object is a subset of the other object.

**Data Architecture**

Data Architecture deals with designing and constructing data resources. Data Architecture provides methodologies to design, construct and implement a fully integrated, business-driven data resource that includes real world objects and events, onto appropriate operating environments. Data Architecture also covers data resource components. Data architecture is one of the pillars of Enterprise Architecture. The other pillars are the Application Architecture, Business Architecture and Integration Architecture. The Data Architecture pillar is the definition or blueprint of the data design which will be used in achieving the implementation of a physical database.

**Data Dictionary**

A centralized repository of logical data elements and other metadata about them. This may include the meaning of a piece of data, relationships to other logical data, origin, usage, type and length. Logical data usually models the real world more closely than physical data since physical data and its structure is usually optimized for system performance purposes.

**Deployment Diagram**

A diagram that helps you visualize how a system will run such as the servers, hardware and software working together. In the diagram below, the system shows how a user (client) accesses a web site and what components must work together for the web site to be delivered to the client:



**Enterprise Architecture**

A multi-dimensional model that attempts to fit together the technologies, information and related activities in alignment with the business service areas. This establishes a common framework for scaling up and down with the right mix of technologies as the business changes. Several models are built separately to form the Enterprise Architecture. For example, you would show major business functions in the Business Architecture and the Information Architecture would show how information is distributed.

**Entity Relationship Diagram**An entity-relationship diagram models the relationships between entities in a database. Standard symbols are used to represent different types of information. The conventional notation uses rectangles to represent entities (nouns), diamonds to represent relationships (verbs) and ovals to represents attributes of entities. Other notations are sometimes used.

**Foreign Key**

A field in a table that links back to another table within a database. The foreign key matches up a column in the child table back to a column in the parent table.

**Gateways**

Similar to a decision diamond in a traditional flowchart, workflows within a diagram are subject to control based on some decision criteria. Depending upon on how the criteria are met, the flow is directed in different ways from the gateway. In the example below, the Gateway is denoted with an X, indicating that it is exclusive; i.e. there is only one path that can be taken. The criteria applied have to do with monetary threshold amounts. For example, if the amount is less than 100, it goes to the “not enough” end event:



**Generalization**

A type of relationship where one object is a specialized object of another. This usually takes the form of an inheritance type relationship. For example, apples, grapes and bananas can be generalized as fruits:



**Multiplicities**

The range of values that an object participates in its relationship to other objects within the model. Many developers refer to this as Cardinality. The number of instances of participation is denoted with symbols, such as the following:

\* - Unlimited number of instances

0..n – From 0 to n number of instances

0..1 - 0 or 1 instances

**Normalization**

A technical procedure of building the right kinds of relationships between objects in databases, such as fields, records, and tables. The goal of normalization is to minimize redundant data in the database by only describing the data as little as possible between tables and at the same time establishing strong relationships between the data through well-organized tables. For example, the first normal rule speaks to elimination of any duplicate columns within the same table.

**Primary Key**

A unique way to identify each record of a database (tables). Each time a record is created, the primary key is created and never changes so that this value remains constant. It usually has very minimal attributes compared to other fields in the database tables.

**Process**

A series of activities that seeks to produce some desired end result. Usually involves role players or people who perform some of the activities. Most processes are repeatable – they are performed over and over again. When a process is repeatable, then you can model its behavior.

**Reinforcing Feedback Loop**

A type of loop used within System Dynamics to show how feedbacks reinforce the accumulation in the model. For example, higher birth rates reinforce and grow population over time. Reinforcing feedbacks can lead to undesirable consequences; almost taking a life on their own until at some point a “balancing” feedback loop kicks in. Therefore, complex systems can easily get out of balance with reinforcing feedback loops.

**Security Architecture**

The infrastructure a company uses for security which can include firewalls, identity management, authentication, vulnerability assessments, and monitoring. Security architecture often represents a layer or dimension within the overall Enterprise Architecture of a company.

**State Chart Diagram**

A type of UML model that starts with an initial state or condition. An object’s state is defined by its attributes and association with other objects. As you modify or change an object’s attributes, its state changes. The State Chart Diagram below tracks the changes that take place with an order for a plane:



Source: [www.sourcemaking.com](http://www.sourcemaking.com)

**Stereo Type**

Elements in a model can have informational tags or captions that imply how the element is used or what kind of object it is. For example, an object may be denoted with the stereo type <<entity>> which means that it is a separate entity within the model and it does not start an interaction within the model. You can also show stereo types within the model, such as how one object is an extension of another:



**Swim Lane Diagram**

A standard framework for describing a process. The main structure of the diagram consists of lanes and pools. Within the lanes are three types of flow objects: Events, Activities, and Gateways. The flow objects can be connected three ways: Sequentially, through Messages or by Association. In the diagram below, a start event kicks off the process on the left side and everything flows sequentially through sub processes or high level activities. After interviews are conducted, a gateway occurs to either hire the candidate or not. The process has an end event after the First Day Briefing activity.



**System**

A large scale collection of processes or sub processes, all working together towards some common goal or objective. All of the elements of a system get described through languages such as the Unified Modeling Language or for highly complex systems, you may have to turn to System Dynamics.

**System Dynamics**

An engineering discipline used to model complex systems based on casual circular relationships that take place per feedback loops. Feedback loops are either balancing or reinforcing within the system. Stocks or Sinks accumulate flows within the model. In the example below, the system dynamics model depicts how the population of sheep and wolves will change given birth rates, death rates, and predation efficiency and rates:



**Unified Modeling Language**

A graphical language used to model systems, developed and released by the Object Management Group. UML is designed to be a standard for how models are expressed for computer applications, not unlike how construction engineers draw up blueprints to develop a building. UML also provides a series of standard diagrams, such as the: Use Case Diagram, Class Diagram, Activity Diagram, Component Diagram, and Deployment Diagram.

**Use Case Diagram**
A use case diagram is a UML diagram that provides a high-level graphical view of the functionality (use cases) supported by the system and shows which roles (actors) can invoke each use case. This high-level view of the system provides a context for the readers of the more detailed use case specifications. See example below for ATM System: 

**Zachman Framework**

A common framework that is useful for designing and understanding a company’s enterprise architecture. The framework will usually take the shape of a 6 x 6 matrix so that all descriptive elements of the enterprise can be defined. An example appears below:

