AGGREGATE

A cluster of associated objects that are treated as a unit for the purpose of data changes. External references are restricted to one member of the **aggregate**, designated as the *root*. A set of consistency rules applies within the **aggregate**'s boundaries.

ANALYSIS PATTERN

A group of concepts that represents a common construction in business modeling. It may be relevant only to one domain or may span many domains.

ASSERTION

A statement of the correct state of a program at some point, independent of how it does it. Typically, an **assertion** specifies the result of an operation of an invariant of a design element.

BOUNDED CONTEXT

The delimited applicability of a particular model. **Bounding Contexts** gives team members a clear and shared understanding of what has to be consistent and what can develop independently.

CLIENT

A program element that is calling the element under design, using its capabilities.

COHESION

Logical agreement and dependence

COMMAND

(a.k.a. *modifier*) An operation that effects some change to the system (for example, setting a variable). An operation that intentionally creates a side effect.

CONCEPTUAL CONTOUR

An underlying consistency of the domain itself, which, if reflected in a model, can help the design accommodate change more naturally.

CONTEXT

The setting in which a word or statement appears that determines its meaning. See **Bounded Context**.

CONTEXT MAP

A representation of the **Bounded Contexts** involved in a project and the actual relationships between them and their models.

CORE DOMAIN

The distinctive part of the model, central to the user's goals, that differentiates the application and makes it valuable.

DECLARATIVE DESIGN

A form of programming in which a precise description of properties actually controls the software. An executable specification.

DEEP MODEL

An incisive expression of the primary concerns of the domain experts and their most relevant knowledge. A deep model sloughs off superficial aspects of the domain and naive interpretations.

DESIGN PATTERN

A description of communicating objects and classes that are customized to solve a general design problem in a particular context.

DISTILLATION

A process of separating the components of a mixture to extract the essence in a form that makes it more valuable and useful. In software design, the abstraction of key aspects in a model, or the partitioning of a larger system to bring the **Core Domain** to the fore.

DOMAIN

A sphere of knowledge, influence or activity.

DOMAIN EXPERT

A member of a software project whose field is the domain of the application, rather than software development. Not just any user of the software, the domain expert has deep knowledge of the subject.

DOMAIN LAYAER

That portion of the design and implementation responsible for domain logic within a **Layered Architecture**. The domain layer is where the software expression of the domain model lives.

ENTITY

An object fundamentally defined not by its attributes, but by a thread of continuity and identity.

FACTORY

A mechanism for encapsulating complex creation logic and abstracting the type of a created object for the sake of a client.

FUNCTION

An operation that computes and returns a result without observable side effects.

IMMUTABLE

The property of never changing observable state after creation.

IMPLICIT CONCEPT

A concept that is necessary to understand the meaning of a model or design but is never mentioned.

INTENTION-REVEALING INTERFACE

A design in which the names of classes, methods and other elements convey both the original developer's purpose in creating them and their value to a client developer.

INVARIANT

An **Assertion** about some design element that must be true at all times, except during specifically transient situations such as the middle of the execution of a method, or the middle of an uncommitted database transaction.

ITERATION

A process in which a program is repeatedly improved in small steps. *Also*, one of those steps.

LARGE-SCALE STRUCTURE

A set of high-level concepts, rules or both that establishes a pattern of design for an entire system. A language that allows the system to be discussed and understood in broad strokes.

LAYERED ARCHITECTURE

A technique for separating the concerns of a software system, isolating a domain layer, among other things.

LIFE CYCLE

A sequence of states an object can take on between creation and deletion, typically with constraints to ensure integrity when changing from one state to another. May include migration of an **Entity** between systems and different **Bounded Contexts**.

MODEL

A system of abstractions that describes selected aspects of a domain and can be used to solve problems related to that domain.

MODEL-DRIVEN DESIGN

A design in which some subset of software elements corresponds closely to elements of a model. *Also*, a process of co-developing a model and an implementation that stay aligned with each other.

MODELING PARADIGM

A particular style of carving out concepts in a domain, combined with tools to create software analogs of those concepts (for example, object-oriented programming and logic programming).

RERPOSITORY

A mechanism for encapsulating storage, retrieval and search behavior which emulates a collection of objects.

RESPONSIBILITY

An obligation to perform a task or know information.

SERVICE

An operation offered as an interface that stands alone in the model, with no encapsulated state.

SIDE EFFECT

Any observable change of state resulting from an operation, whether intentional or not, even a deliberate update.

SIDE-EFFECT-FREE FUNCTIONS

See **function**.

STANDALONE CLASS

A class that can be understood and tested without reference to any others, except system primitives and basic libraries.

STATELESS

The property of a design element that allows a client to use any of its operations without regard to the element's history. A stateless element may use information that is accessible globally and may even change that global information (that is, it may have side effects) but holds no private state that affects its behavior.

STRATEGIC DESIGN

Modeling and design decisions that apply to large parts of the system. Such decisions affect the entire project and have to be decided at team level.

SUPPLE DESIGN

A design that puts the power inherent in a deep model into the hands of a client developer to make clear, flexible expressions that give expected results robustly. Equally important, it leverages that *same* deep model to make the design itself easy for the implementer to mold and reshape to accommodate new insight.

UBIQUITOUS LANGUAGE

A language structured around the domain model and used by all team members to connect all the activities of the team with the software.

UNIFICATION

The internal consistency of a model such that each term is unambiguous and no rules contradict.

VALUE OBJECT

An object that describes some characteristic or attribute but carries no concept of identity.

WHOLE VALUE

An object that models a single, complete concept.