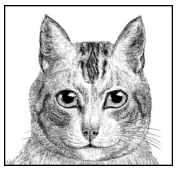
,ch02.17200 Page 15 Wednesday, May 9, 2001 4:00 PM



CHAPTER 2

The Big Picture

This chapter introduces the context for understanding and applying the UML. After reading this chapter, you will understand the following:

- An overall view of problems, solutions, and problem solving.
- Life-cycle concepts, including development cycles and phases and iteration cycles and phases.
- Problem and solution concepts, including domains or spaces, systems, architectures, models, architectural views, diagrams, and languages.
- Problem-solving concepts, including paradigms, artifacts, activities, and heuristics.

This chapter introduces some of the key concepts underlying the UML. I start by introducing the whole notion of problem solving, which involves how an effort is distributed over a life cycle in order to solve a problem. I then introduce the details of how we understand problems and solutions. I also introduce the details of how we understand problem-solving processes.

Problems, Solutions, and Problem Solving

Organizations produce and deliver products and services that address customer needs and requirements. Requirements may be characterized as *problems*. Products and services that address requirements are characterized as *solutions*. To deliver valued solutions (maximum quality and minimum cost within the minimum time), organizations must capture (acquire), communicate (share), and leverage (utilize) knowledge. The value of a solution is determined by the quality of the product or service, the cost of the product or service, and the time needed to produce the product or deliver the service. These producer-consumer relationships exist at all levels within the professional world: Organizations solve problems for their customers, employees solve problems for their employers, doctors solve problems for their patients, and so forth. Within these types of relationships, knowledge is the determining factor of success.

Organizations use technology to disseminate and communicate information so that their employees can solve business problems. Technology is only a tool; without people knowing how to apply it and actualize its potential, it is useless. Because the business world and technology world are in constant flux, we are required to manage increasing complexity within our ongoing efforts. However, we ought not to forget that technology is only a means to an end rather than the end in itself.

Projects are problem-solving efforts. They involve understanding or conceptualizing a problem, solving the problem, and implementing or realizing the solution. The people or organizational entities involved in a project are known as *stakeholders*. Some stakeholders are the ones for whom the problem is solved. They are responsible for contributing their knowledge to understand the problem and verify the solution. Other stakeholders are responsible for contributing their knowledge to implement the solution. The results of a project are know as *deliverables* or *work products*. They are the products or services that solve a problem. Projects, fundamentally, formalize the "work hard and hope for the best" approach to problem solving.

The UML is a language that facilitates the following:

- Specifying, visualizing, understanding, and documenting problems.
- Capturing, communicating, and leveraging knowledge in problem solving.
- Specifying, visualizing, constructing, and documenting solutions.

However, the UML does not prescribe any particular problem-solving approach. Rather, it is very flexible and customizable to fit any approach.

Programs are collections of problem-solving efforts (or projects). They are coordinated and related so that each project may benefit from the work of other projects in the program. One of the critical criteria necessary for achieving this objective is communication. If communication is possible across projects, it may be more likely that work and knowledge will be readily shared and work effort reduced. The role of the UML is to enable and facilitate such communication.

Consider an organization that is having trouble optimally allocating its employees to various projects. It desires an information system for maintaining employee and project information. The solution should provide the capability to allocate employees to projects based on their skills and where they may be best utilized. In addressing this problem, an overall approach must address how we will understand or conceptualize the problem, derive a solution to the problem, and implement or realize the solution. This approach will determine how we view the problem (paradigm) for the purpose of understanding it and how we view the resulting solution (paradigm) for the purpose of realizing it. We will apply our knowledge of the situation and other rules of thumb (heuristics) gained from other experiences to derive the solution (artifacts). Our effort will be organized (life cycle) as a series of (possibly concurrent) steps (activities) so that it may be managed to develop the resulting information system. We examine these constructs in the section "Problem Solving."

The problem occurs within a business context (domain or space). The solution must be realized to fit within the organization's information technology infrastruc-

ture (domain or space). The business problem (system) must be fully understood in terms of the requirements of the business, and the information system must be fully understood in terms of how it meets those requirements. The resulting information system must also be organized (architecture) to fit into the organization's technology infrastructures. As we conceptualize the problem and work toward realizing the solution, we will capture knowledge (models) about the business problem and information system, make decisions (architectural views) about how we will address different issues, and should be able to depict and communicate this information (diagrams). We examine these constructs in the section "Problems and Solutions."

Methods specify how to conduct problem-solving efforts. They specify an overall problem-solving approach and its components. They specify how problems and solutions are viewed in relation to a problem-solving approach; this is known as a method's *descriptive* aspect since it describes how knowledge is captured and communicated regarding a problem and solution. Methods also specify a problem-solving approach to be used to solve the problem and derive a solution; this is known as a method's *prescriptive* aspect since it prescribes how knowledge is leveraged to solve a problem. Methods specify descriptively how problems and solutions are viewed, and prescriptively how the problem-solving effort may be actualized.

Processes are realizations of methods. They use a method's problem-solving approach to solve a problem and realize a solution. Processes are specified by methods; this is known as a process's *static* aspect since it is a description of how a problem may be solved. Processes apply methods within projects to solve problems; this is known as a process's *dynamic* aspect.

Since projects can be very complex, methods should be used to provide an infrastructure for the problem-solving process. They should be considered as suggestions and recommendations that organize and facilitate the problem-solving process rather than being considered rigid and inflexible rules that restrict the art of problem solving. A *methodology* is a taxonomy, or well-organized collection, of related methods; therefore, a method may be part of a methodology.

To understand how the UML can be used to facilitate problem solving, a deeper understanding of what really constitutes problems, solutions, and problem solving is required. Such an understanding will enable you to analyze a problem and determine what parts of the UML may be best applied to facilitate deriving the solution to the problem.

Problems are often referred to as *as-is* situations since they represent the current conditions; solutions are similarly referred to as *to-be* situations since they represent the target conditions we are trying to reach. See Figure 2.1.

A situation that does not cause difficulty may still be treated as a problem in order to understand it; however, the solution is nothing more than an understanding of the situation. This results in determining the characteristics, rather than the requirements, of the situation and provides a purely descriptive view of the situation without any descriptive or prescriptive view of a solution. This is often done when a solution to a problem exists, and the existing solution needs to be understood or evaluated.

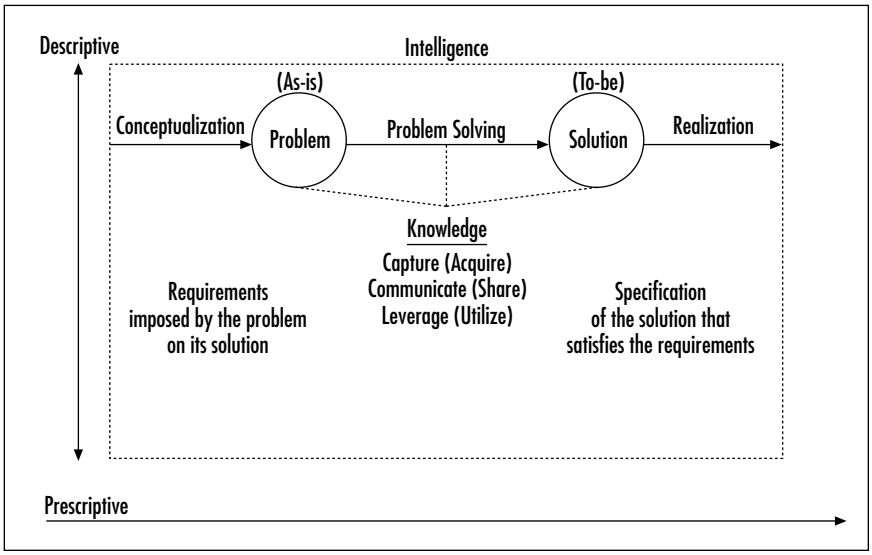


Figure 2-1: Overall View of Problem Solving

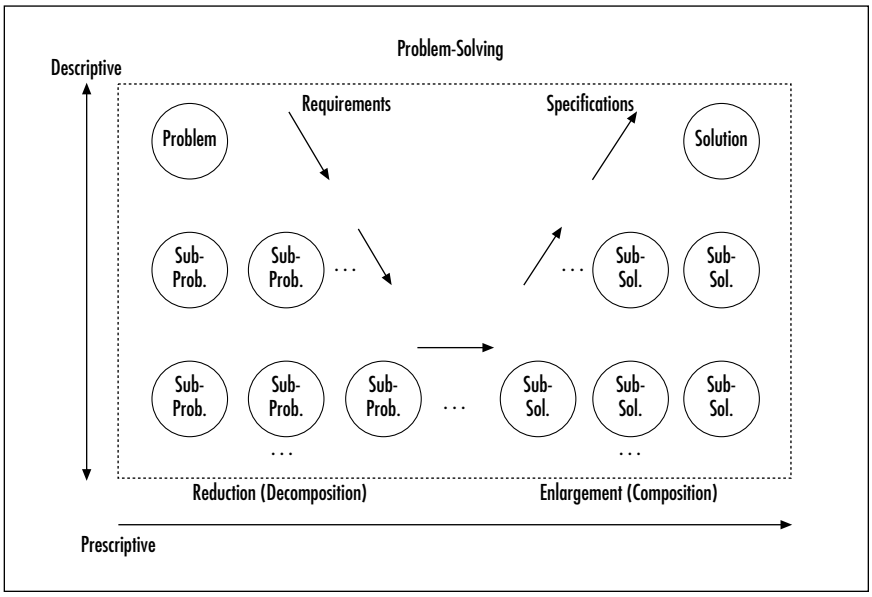


Figure 2-2: Detailed View of Problem Solving

Problem solving (Figure 2-2) involves efforts or processes that cooperatively solve problems. Such efforts include the following:

- Conceptualizing and representing the problem in some malleable and communicable form. This allows the problem to be represented in some language so that it may be manipulated and communicated to others.

- Determining the requirements imposed by the problem on its solution, that is, the requirements caused by the problem that its solution must satisfy.
- Attaining a description of the problem by addressing the following questions: What is the problem? What are the requirements of the problem that its solution must satisfy? Within a project, if these questions can be minimally answered with a sufficient level of detail, the effort has attained a descriptive view of the problem.
- Identifying or specifying the solution that satisfies the requirements imposed by the problem.
- Attaining a description of the solution by addressing the following questions: What is the solution? What are the specifications of the solution that satisfies the requirements imposed by the problem? Within a project, if these questions can be minimally answered with a sufficient level of detail, the effort has attained a descriptive view of the solution.

If a solution cannot be identified for a problem, the requirements imposed by the problem on its solution are grouped, classified, and organized into subordinate problems that are solved concurrently. A superordinate problem may be recursively reduced or decomposed into subordinate problems. Subordinate problems may be solved concurrently, and the solution to the original problem may be recursively enlarged or composed from subordinate solutions. The reduction process continues until all the subordinate problems are solved, and the enlargement process continues until the solution to the original problem is derived. Because this process is recursive, the descriptive view is used to discover the prescriptive view. Furthermore, decomposition allows a great deal of concurrent and parallel problem solving so that the whole process is expedited.

Problem solving continues as follows:

- Attaining a prescription of the steps within the solution that solves the problem by addressing the following question: How does the solution, in turn, have its requirements (or subordinate problems) satisfied? Within a project, if these questions can be minimally answered with a sufficient level of detail, the effort has attained a prescriptive view of the solution.
- Realizing the representation of the solution in some concrete and usable form.

Knowledge is the overriding component that must be captured (acquired), communicated (shared), and leveraged (utilized) to facilitate this process. The total knowledge involved in such an effort may be segmented into groups, where each group is associated with some part of the effort (conceptualization, problem, solution, problem solving, or realization) at any given point in time. This association signifies when the knowledge elements influence the effort; that is, when the knowledge elements were acquired, shared, or utilized within the problem-solving effort.

Intelligence is the ability to conduct the problem-solving process by organizing knowledge in a taxonomy and strategically applying knowledge via a problem-solving effort in order to derive a solution.

Life Cycle

Problem-solving approaches are organized (life cycles) to offer a management perspective and a development perspective. The two perspectives enable the effort to be managed and performed. The life cycle will be discussed again in Chapter 5 as it relates to the UML.

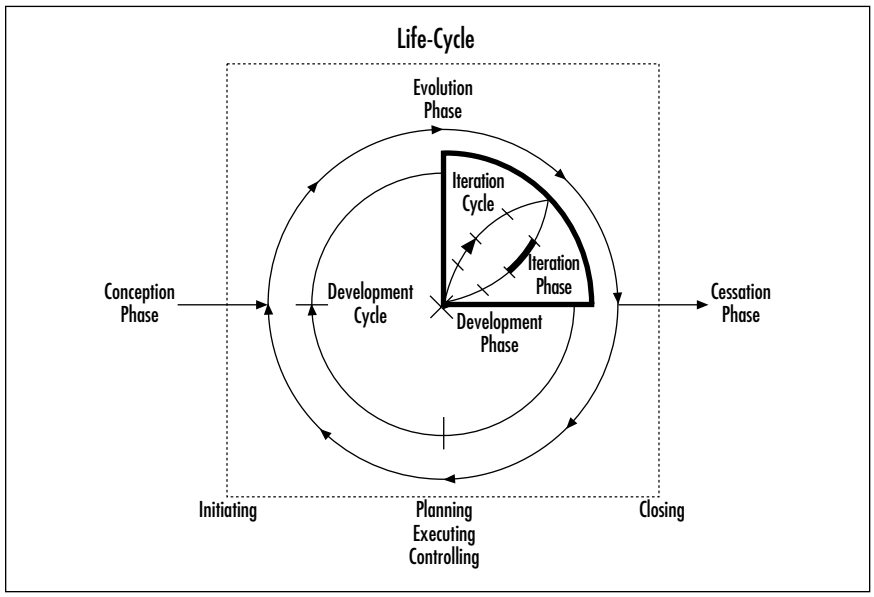


Figure 2-3: Life Cycle

Life cycles (Figure 2-3) can be broken down as follows.

- A *life cycle* is a collection of phases that divide an effort into several more manageable and controllable subordinate efforts.
- Each phase within a life cycle is a discrete part of a single flow of logic to solve a problem. Each phase
  - Has inputs, or items that will be acted on. This addresses what is manipulated in the phase.
  - Has entry criteria for entering or initiating the phase.
  - Requires the application of principles, techniques, and tools to the inputs to generate the outputs. This addresses how something is manipulated and who is responsible for doing the manipulation.
  - Has outputs, or items that result from the phase's effort.
  - Has exit criteria for exiting or closing the phase. This results in a review of the resulting work products and, often, a decision whether to continue or abandon the overall effort.
- The conception phase involves recognizing a problem or opportunity and committing to solving the problem.

- The evolution phase includes multiple evolutionary cycles of initiating, planning, executing, controlling, and closing an effort.
  - Initiating involves recognizing that an evolution cycle is required in addressing a new or existing problem.
  - Planning involves designing and maintaining a scheme to accomplish the solving of the problem.
  - Executing involves coordinating resources for carrying out the plan and solving the problem.
  - Controlling includes monitoring and measuring progress so that corrective measures may be taken when necessary.
  - Closing involves formalizing acceptance of the current cycle and the solution to the problem for which the cycle was initiated.

  The evolution phase consists of one or more development cycles (evolution cycles), where each development cycle includes a single effort distributed among the constituents of the development cycle. It encompasses the evolution of a product as it grows and matures throughout its existence.
- A development cycle
  - Consists of development phases.
  - Distributes effort among its development phases.
  - Encompasses one evolutionary generation of a product in its life-cycle evolution.
  - Occurs the first time in response to a problem and produces a solution or product.
  - Occurs multiple times thereafter to help an existing solution or product evolve.
  - Is triggered by the occurrence of a problem and the need for a solution or by changes in a problem and the need for evolution in an existing solution. These changes may include changes in stakeholder requirements, underlying technology, and organizational competitiveness.
  - May slightly overlap with its predecessor or successor development cycles. The start of the current development cycle may slightly overlap with the end of the previous development cycle, and the end of the current development cycle may slightly overlap with the start of the next development cycle.
  - Produces a product generation.
- A development phase
  - Consists of iteration cycles.
  - Distributes effort among its iteration cycles.
  - Encompasses one evolutionary phase of a product in its development-cycle evolution.

  - Is triggered by the start of a new development cycle or the completion or near completion of the previous development phase within a development cycle.
  - May slightly overlap with its predecessor or successor development phases. The start of the current development phase may slightly overlap with the end of the previous development phase, and the end of the current development phase may slightly overlap with the start of the next development phase.
  - Synchronizes the development-oriented elements and management-oriented elements via milestones, that is, points in a development cycle where critical decisions must be made and key goals achieved.
- An iteration cycle
  - Consists of iteration phases.
  - Distributes effort among its iteration phases.
  - Encompasses one evolutionary iteration of a product in its development-phase evolution.
  - Is triggered by the start of a new development phase or the completion or near completion of the previous iteration cycle within a development phase.
  - May slightly overlap with its predecessor or successor iteration cycles. The start of the current iteration cycle may slightly overlap with the end of the previous iteration cycle, and the end of the current iteration cycle may slightly overlap with the start of the next iteration cycle.
  - Produces a product release. This is a subset of a complete product generation.
- An iteration phase
  - Consists of activities or tasks.
  - Distributes effort among its activities or tasks.
  - Encompasses one evolutionary increment of a product in its iteration-cycle evolution.
  - Is triggered by the start of a new iteration cycle or the completion or near completion of the previous iteration phase within an iteration cycle.
  - May slightly overlap with its predecessor or successor iteration phases. The start of the current iteration phase may slightly overlap with the end of the previous iteration phase, and the end of the current iteration phase may slightly overlap with the start of the next iteration phase.
- The *cessation phase* involves recognizing that a solution is no longer required and terminating the application of the solution in an orderly manner.
- A great deal of concurrency and parallelism occurs within a life cycle; thus the overall process is expedited as an effort or project is distributed over a development cycle.

Development Cycles and Phases

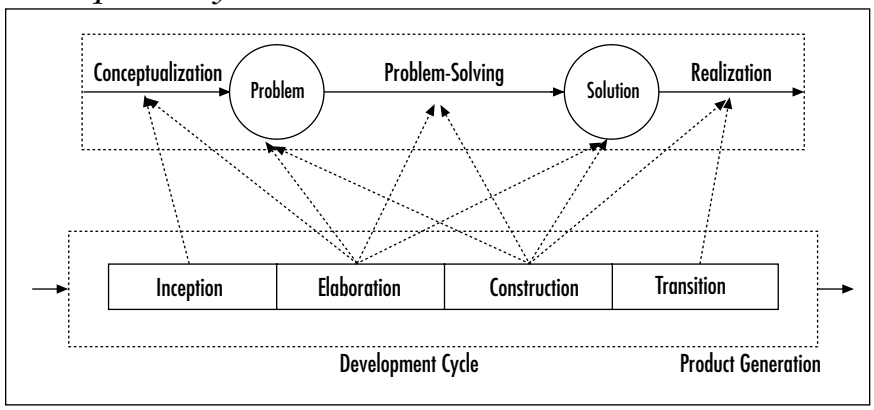


Figure 2-4: Focus of Development Phases within a Development Cycle

Development cycles and phases (Figure 2-4) have the following characteristics:

- Each development cycle consists of four phases (inception, elaboration, construction, and transition) and results in a product generation. This view of the life cycle is often called the *management perspective* since it provides high-level indicators of a project's progress.
- The inception (initiating) development phase usually uses one iteration cycle and involves the following activities:
  - Specify and delimit scope, objectives, and requirements.
  - Establish a business case or rationale, including specifying the vision, specifying success criteria, assessing risks, estimating time and cost, and specifying a plan.
  - Focus on understanding or forming a notion of the problem and the rationale for solving it (scoping and business case).
- The elaboration (planning) development phase usually uses one iteration cycle and involves the following activities:
  - Elaborate the specification of the effort from the previous phase.
  - Baseline and completely delimit scope, objectives, and requirements.
  - Baseline the business case by solidifying the vision, solidifying success criteria, and mitigating the highest risks. Baseline the plan with a schedule.
  - Distribute the requirements among multiple iteration cycles within the construction phase.
  - Focus on understanding or forming a notion of the problem to determine the requirements that the problem imposes on its solution (requirements capture and requirements analysis), establishing and verifying the foundation for the overall solution (architectural design), and

distributing the requirements among the iteration cycles of the construction development phase (planning).

- The construction (executing) development phase usually uses more than one iteration cycle and involves the following activities:
  - Update the business case by managing ongoing risks and maintaining the plan and schedule.
  - Build and develop the effort's products by performing the multiple iteration cycles.
  - Focus on the following: understanding or evolving the requirements that the problem imposes on its solution (requirements) so that the problem can be elaborated and the solution can be specified, elaborating the solution specification (analysis), updating the foundation for the overall solution (architectural design) and the foundation required to support the specific solution (detailed design) for the iteration cycle requirements, generating the solution (implementation) for the iteration cycle requirements, verifying the solution (validation and integration) for the iteration cycle requirements, and providing or integrating (or delivering) the solution (deployment) or a subset thereof.
- The transition (closing) development phase usually uses one iteration cycle and involves the following activities:
  - Deploy the effort's ultimate product or solution to the problem.
  - Assess and classify the effort's interim products.
  - Focus on providing and integrating or delivering the solution (deployment).

Iteration Cycles and Phases

Iteration cycles and phases (Figure 2-5) have the following characteristics:

- Each iteration cycle consists of two support components (management and support) and six development components (requirements, analysis, design, implementation, validation, and deployment). It results in a product release. This view of the life cycle is often called the *development perspective* since it provides low-level indicators of a project's progress.
- The management (controlling) phase involves managing the overall iteration cycle.
- The support (controlling) phase involves supporting the needs and requirements of the stakeholders participating in the iteration cycle; this includes development environment support and configuration, standards definition, and guidelines.
- Within the elaboration (planning) development phase, the following iteration phases apply:
  - The requirements (requirements capture) and analysis (requirements analysis) phases involve understanding or forming a notion of the problem to determine the requirements of the solution and distributing the require-

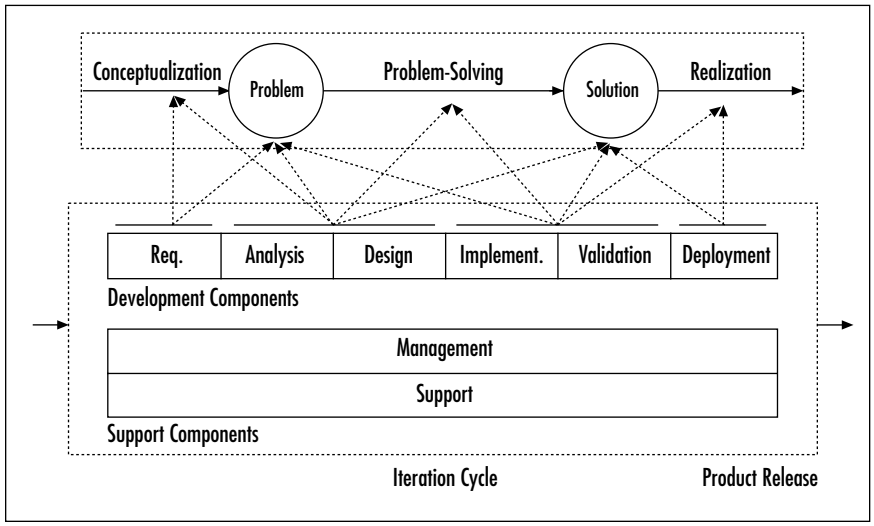


Figure 2-5: Focus of Iteration Phases within an Iteration Cycle

ments among iteration cycles of the construction development phase (planning).

  - The design (architectural design) phase involves establishing and verifying the foundation for the overall solution.

- Within the construction (executing) development phase, the following iteration phases apply:
  - The requirements phase involves understanding or elaborating the requirements the problem imposes on its solution so that the problem can be elaborated and the solution can be specified.
  - The analysis phase involves elaborating the solution specification. This phase hinges on the application of knowledge.
  - The design phase involves updating the foundation for the overall solution (architectural design) and the foundation required to support the specific solution (detailed design) for the iteration cycle requirements. This phase hinges on the application of knowledge.
  - The implementation phase involves generating the solution for the iteration cycle requirements.
  - The validation (and integration) phase involves verifying the solution for the iteration cycle requirements.
  - The deployment phase involves providing and integrating or delivering the solution or a subset thereof.

Problems and Solutions

Problems and solutions (Figure 2-6) occur within a context (domain or space). The problem (system) must be understood in order to be solved. The solution (system)

to a problem must be understood in order to be constructed and used. The solution must be organized (architecture) in order to facilitate its realization and adhere to the various constraints of the context in which it will be realized. To solve the problem, appropriate knowledge about the problem and solution must be captured (models), organized around decisions regarding the problem and solution (architectural views), and depicted (diagrams) using some language that enables it to be communicated and leveraged in the problem-solving process.

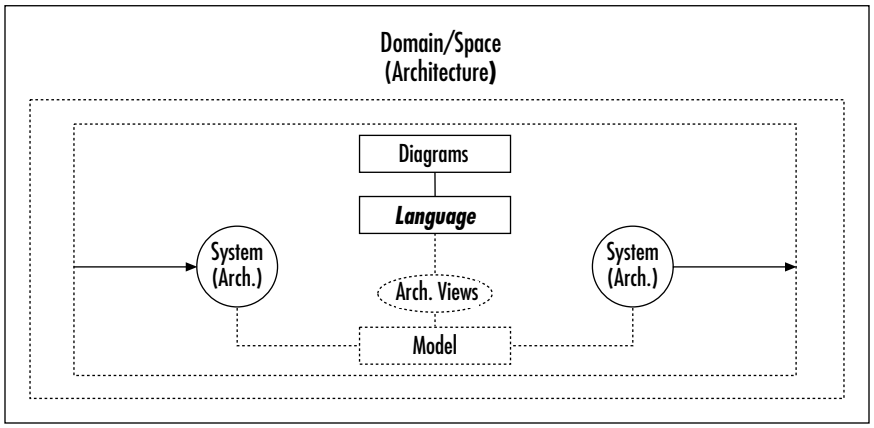


Figure 2-6: Problems and Solutions

Domains or Spaces

Domains or spaces (Figure 2-7) are organized collections of related components in a self-contained situation or area of interest. These components are called *domain* or *space elements*. They

- Are also known as contexts.
- Have a set of terms forming a vocabulary that facilitates communication regarding the components within the domain.
- Have a set of concepts forming ideas or notions associated with components within the domain and giving meaning to the communications regarding the components of the domain.
- Have a definitive and delimiting boundary defining the context of scope for the domain. This identifies an inner realm where every component inside the boundary is within the context and every component outside the boundary is in an outer realm or some other context.
- Have no purpose. That is, the components that constitute a domain are not organized to accomplish a set of objectives, but are incidentally organized together for some other reasons or basis. Often, the components of a domain are organized together simply because they relate to the same subject of the domain.
- May represent real or abstract situations.

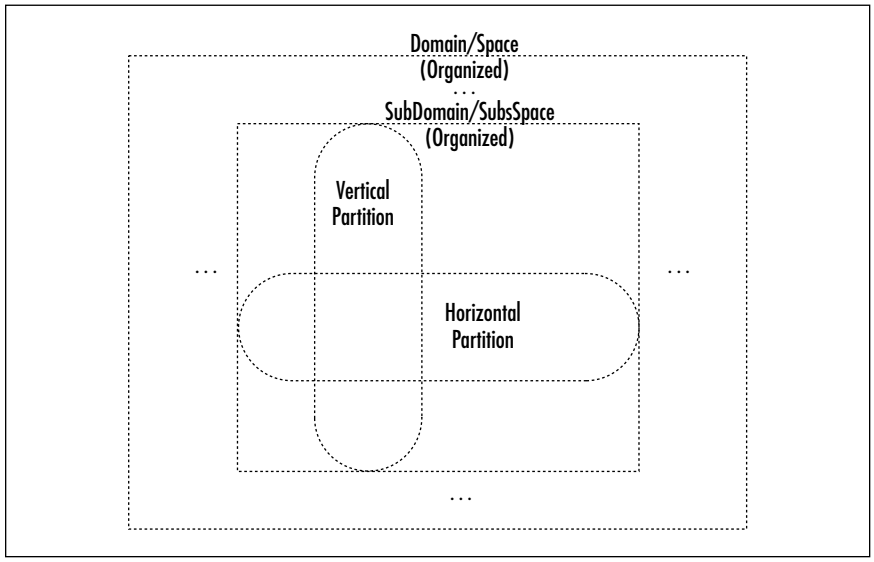


Figure 2-7: Domains or Spaces

- May be recursively decomposed into multiple domains called *subordinate domains*. These subordinate domains may be considered topics of interest within the domain's subject.
- May be decomposed, segmented, and viewed vertically or in vertical partitions of components as multiple areas of interest and concerns.
- May be decomposed, segmented, and viewed horizontally or in horizontal partitions of components as crossing multiple areas of interest and concerns. These partitions reveal progressively more detailed information with depth.
- May be the subject of multiple bodies of knowledge, where each body of knowledge may have different content yet is derived by focusing on the same domain.
- Represent contexts or frameworks in which problems and solutions exist and are addressed.

The UML is domain independent. It may be utilized within domains to facilitate problem solving, and across domains to facilitate the reuse of knowledge. In developing information systems, domains are the contexts in which problems exist and solutions are realized.

Systems

Systems (Figure 2-8) are organized collections of interacting and connected components cooperating to accomplish a purpose. These components are called *system elements*. Systems

- Are contexts that have a purpose, goal, or responsibility. Systems may be treated as contexts if their purposefulness is not utilized or is simply ignored, and contexts may be treated as systems by introducing a purpose. Contexts

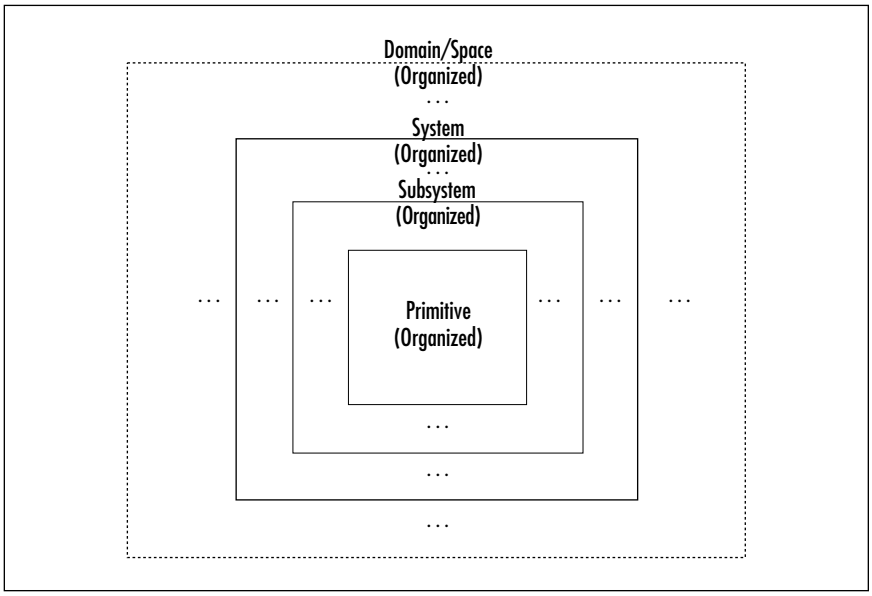


Figure 2-8: Systems

and systems are the same, differing only when treated or manipulated on the basis of their purpose or on the basis of having a purpose.

- Must have a purpose or set of purposes. That is, they are organized to accomplish a set of objectives, and are essentially organized by virtue of their objectives or goals.
- May represent real or abstract purposeful entities.
- May be recursively decomposed into multiple systems called *subordinate systems* (or *subsystems*).
- When fully decomposed, consist of primitive system elements that are not further decomposable.
- May be the subject of multiple bodies of knowledge, where each body of knowledge may have different content yet is derived by focusing on the same system.
- Represent problems and solutions that exist and are addressed within a context.

The UML is system independent. It may be utilized to address different types of systems, including hardware systems, software systems, and human interaction systems. In developing information systems, systems are the realized solutions to problems.

Architectures

Architectures (Figure 2-9) are schemes involving the structural and behavioral organization of systems within contexts. Architectures

- Have an inward focus on systems.

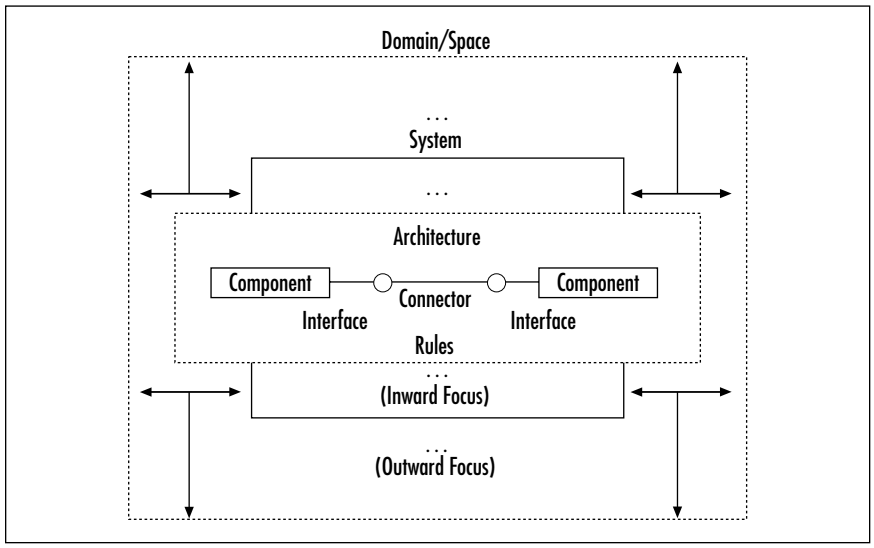


Figure 2-9: Architectures

- Have an outward focus on contexts.
- Include components and their interfaces. These are systems and subordinate systems. The components are called *architectural elements*.
- Include connectors that integrate and connect other components via their interfaces. These connectors are also called *architectural elements*.
- Include rules and constraints governing the assembly of components and connectors.
- Include rules and constraints governing the interactions among components and connectors.
- Involve the arrangement of components connected via connectors in order to satisfy some purpose.
- May be recursively decomposed into multiple architectures called subordinate architectures. Each component and connector may be recursively decomposed into successively smaller components and connectors.
- May be decomposed, segmented, and viewed in vertical partitions of components as subordinate architectures of interest.
- May be decomposed, segmented, and viewed in layered horizontal partitions of components that cross subordinate architectures of interest.
- When fully decomposed, consist of primitive components and primitive connectors that are not further decomposable. The primitive components and primitive connectors correspond to primitive system elements.
- May be classified as forms or ready-made patterns that solve recurring architectural problems. These patterns capture significant knowledge that may be reused.

- May be classified as architectural styles that reference a collection of forms or patterns. These styles are used to impose a degree of uniformity on architecture selection.
- Involve the organization of problems and solutions. They are very well suited for managing overwhelming complexity.
- Represent the internal organization of problems, solutions, and the contexts in which they exist and are addressed.

The UML is architecture-centric. It supports the organization of systems within domains. Organizational mechanisms address and attempt to minimize the complexity that arises from changes to systems. In developing information systems, a well-organized system allows changes to the system to be less time consuming and less costly than an unorganized system does.

Models

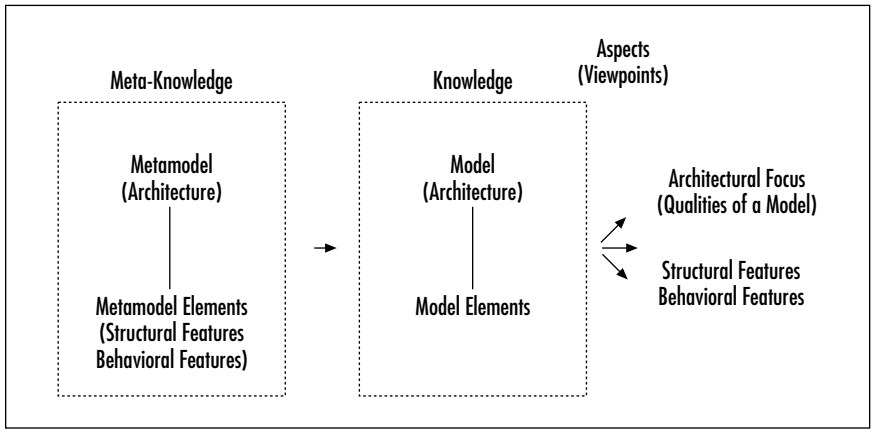


Figure 2-10: Models

Models (Figure 2-10) are complete abstractions of systems or contexts. Models

- Are complete or semantically closed; that is, comprehending and interpreting the contents of a model does not require the introduction of other information from outside the model.
- Are abstract or "fictitious" constructs that represent systems or contexts as the given subject of a model.
- Utilize abstraction to focus on the relevant details of a system while ignoring insignificant details. This defines a boundary of a system relative to the perspective of a viewer.
- Are blueprints of systems used for system construction and renovation.
- Are used to understand and manage complexity within systems.
- Are used for communication and assurance of architectural soundness.
- May be recursively decomposed into multiple models called *subordinate models*.

- When fully decomposed, consist of model elements that are primitive and not further decomposable.
- Have elements as atomic constituents. Model elements are complete abstractions of system elements or context elements. They
  - Are defined by modeling elements. Metamodels are models that define the modeling elements (metamodel elements) that may be used within derived models. Models are instances, occurrences, or configurations of metamodels that contain model elements. Model elements are instances, occurrences, or configurations of modeling elements (metamodel elements).
  - Capture features or characteristics of system elements.
  - Capture the structural features of system elements, that is, the static or structural features of system elements.
  - Capture the behavioral features of system elements, that is, the dynamic or behavioral features of system elements and interactions or collaborations among system elements.
  - May be relationships that relate other elements within models.
- May be viewed via a small set of holistic but nearly independent and nonoverlapping aspects. Aspects are dimensions of modeling that emphasize particular qualities of a model. The following aspects exist for any model:
  - The structural model aspect emphasizes the static, or structural, features of the model elements within a model.
  - The behavioral model aspect emphasizes the dynamic, or behavioral, features of the model elements and interactions or collaborations among model elements within a model.
- May be expressed at different levels of fidelity or accuracy.
- May be reference models that describe all possible configurations of their model elements. Architectures are subsets of reference models. Implementations are products that result from implementing specified architectures.
- Have an architecture when treated as systems whose purpose is to capture a complete abstraction of other systems.
- Capture knowledge regarding a system or context.
- Represent knowledge of problems, solutions, and the contexts in which they exist and are addressed.

The UML enables capturing and leveraging knowledge to facilitate problem solving. Knowledge cannot explicitly be viewed; however, it can be demonstrated. A demonstration of knowledge occurs in the application of knowledge to derive a solution to a problem. This is known as intelligence.

Architectural Views

Architectural views (Figure 2-11) are abstractions of models. Architectural views

- Are perspectives through which models may be represented or projected as diagrams.

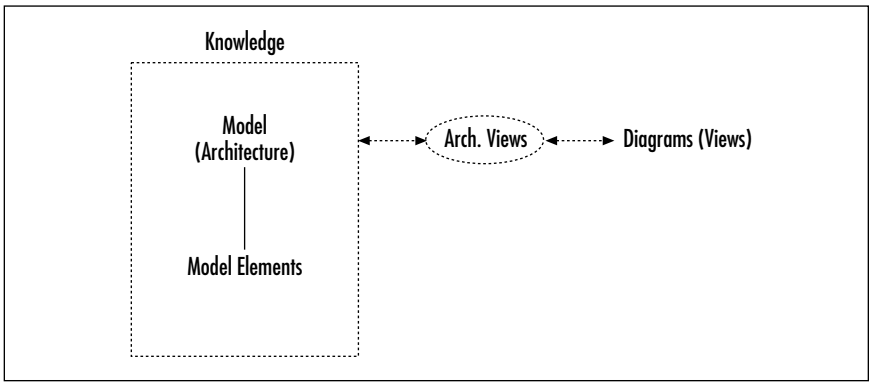


Figure 2-11: Architectural Views

- Organize perspectives or views of models around specific sets of concerns particular to different stakeholders involved in the problem-solving effort. A set of concerns establishes an architectural focus.
- Enable the extraction of architecturally significant elements of a model via different viewpoints or perspectives. Architecturally insignificant elements of a given perspective are omitted for that perspective. A set of aspects of a model that addresses an architectural focus establishes a viewpoint.

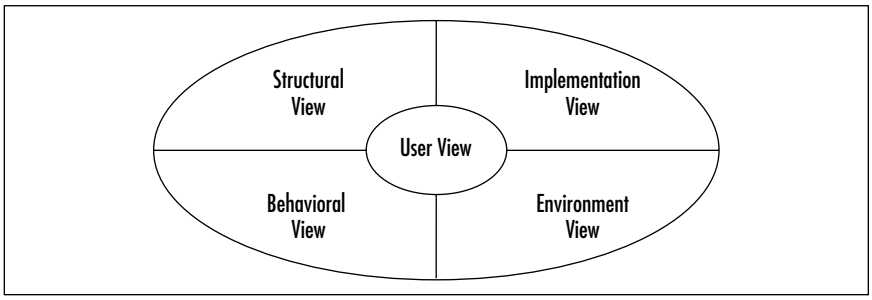


Figure 2-12: Model Views

- Include the following model views (Figure 2-12) regarding models of problems and solutions:
  - The *user model view* encompasses a problem and solution from the perspective of those individuals whose problem the solution addresses. This view presents the goals and objectives of the problem owners and their requirements of the solution.
  - The *structural model view* encompasses the static, or structural, aspects of a problem and solution.
  - The *behavioral model view* encompasses the dynamic, or behavioral, aspects of a problem and solution and interactions or collaborations among problem and solution elements.

  - The *implementation model view* encompasses the structural and behavioral aspects of the solution's realization.
  - The *environment model view* encompasses the structural and behavioral aspects of the domain in which a solution must be realized.
- May be extended to include other views of models (such as interface, security, and data).
- Are quite often referred to as models because they ultimately map back to a subset of a whole model.
- Map models to types of diagrams.
- Organize knowledge within a model around decisions concerning the knowledge and subject. These decisions further constrain the knowledge and future decisions.

The UML supports the user model view, structural model view, behavioral model view, implementation model view, and environment model view to facilitate the organization of knowledge within problem solving.

Diagrams

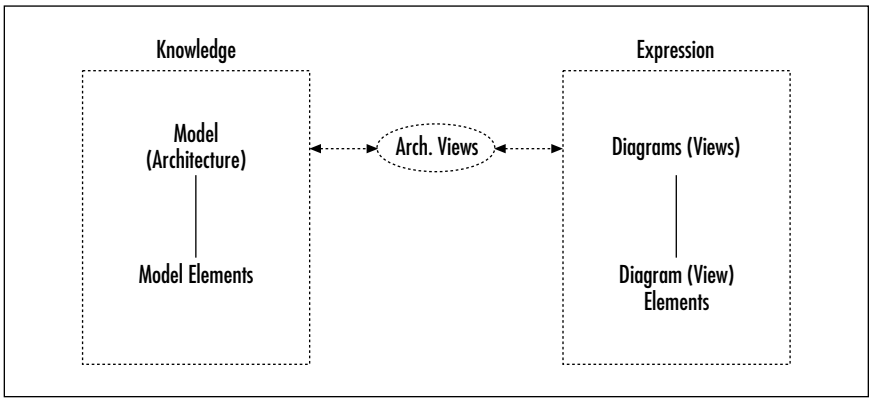


Figure 2-13: Diagrams

Diagrams (Figure 2-13) are graphical projections of sets of model elements. Diagrams

- Are connected graphs of arcs (or paths) and vertices (or nodes).
- Render architectural views or subsets of architectural views of a model pictorially or graphically.
- Are collections of graphical constructs with visual relationships.
- Are concrete and real constructs describing models.
- Are blueprints of models described via architectural views.
- Have view elements that are textual or graphical projections of collections of model elements.
- Are used to communicate model content or knowledge.

- Depict knowledge within a model in a communicable form.
- Are also called models because they render models or are the visual representation of models.

The UML enables communication of knowledge to facilitate problem solving by supporting a multitude of diagrams organized around architectural views.

Languages

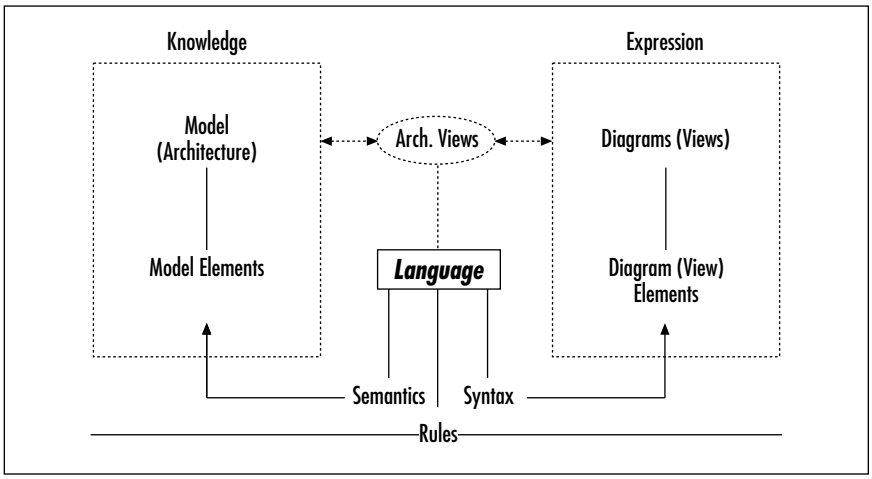


Figure 2-14: Languages

Languages (Figure 2-14) are means for expressing and communicating content or information. Languages

- Have concepts and semantics that are communicated. In the UML, model elements define the fundamental modeling concepts and semantics that establish the meaning or content of communication.
- Have a notation or syntax used to render semantics for communication. Syntactic elements represent semantic elements. In the UML, the notation or visual rendering of model elements provides the syntax and establishes the expression of the semantics for communication.
- Have rules or guidelines expressing idioms of usage and how syntactical constructs are combined in forming the content of communication. The UML captures various guidelines or idioms of usage and best practices.
- Map one or more model elements to one or more diagram elements.
- Such as the UML govern how knowledge within a model is depicted so that it can be communicated.

Problem Solving

Problem solving (Figure 2-15) requires being able to view the problem (paradigm) for the purpose of understanding it, and being able to view the resulting solution

(paradigm) for the purpose of realizing it. The problem-solving process involves leveraging knowledge to derive the solution (artifacts) to the problem through a series of (possibly concurrent) steps (activities) in which knowledge is applied and various tools are utilized. Furthermore, knowledge and rules of thumb (heuristics) gained from other problem-solving efforts may be used.

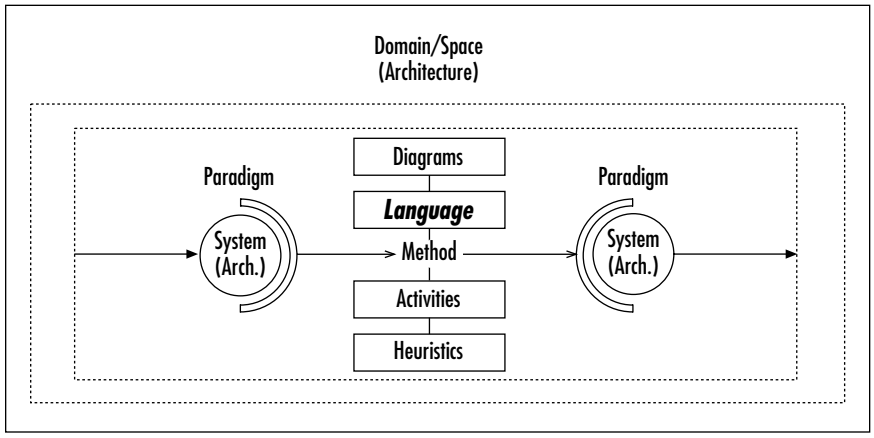


Figure 2-15: Problem Solving

Paradigms

Paradigms are organized, self-contained collections of related components that form the bases for models. These components are called *paradigm elements*. Paradigms

- Define a set of terms forming a vocabulary.
- Define a set of concepts forming ideas or notions associated with subjects.
- Establish the foundation for metamodels by defining the modeling elements that may be used in modeling. They define the terms and concepts that may be used within models.
- Form the foundation for languages and methods. The UML is founded on the object-oriented paradigm and may be used by different methods.
- Establish concepts and associated terms that are fundamental, essential, and universally accepted truths or maxims (principles) within the paradigm. These concepts form the foundation upon which other concepts are built and through which systems are modeled, viewed (perceived, understood, and interpreted), and manipulated.
- Represent how a method approaches, views, and models problems, solutions, and the contexts in which they exist and are addressed.

The UML is founded on the object-oriented paradigm. This enables the UML to be flexible for use by a multitude of methods across various domains concerning various types of systems. The object-oriented paradigm also offers other benefits for managing change and complexity.

Artifacts

Artifacts are work products or deliverables resulting from efforts. Artifacts

- Are specified or described by methods.
- Are realized or constructed within processes.
- Must have an order in which they are created.
- Must have stakeholders responsible for producing them and others responsible for accepting (or rejecting) them.
- Are expressed in languages such as the UML. This expression enables the realization of the solution to a problem.
- Become assets if they may be reapplied or reused in the same context but with different systems or across different contexts and systems.
- Represent the interim and ultimate products (solution) as specified by a method and realized by a process.

The UML diagrams are artifacts that facilitate problem solving. They organize and depict required information for realizing a solution to a problem.

Activities

Activities are efforts or collections of tasks directed at producing or developing artifacts. Activities

- Are specified and partially or tentatively ordered by methods.
- Are realized within processes.
- Must have associated stakeholders who conduct or perform the tasks.
- Must have associated artifacts that they develop (create, modify, etc.).
- Must address both technical and managerial (human) issues related to an effort.
- Must be treated as sets of recommendations and must be customized to the context in which they are applied.
- Establish or recommend tasks and techniques in order to accomplish objectives. *Tasks* specify thinking, performing, and reviewing actions to meet the goals of an activity. *Techniques* provide the means for performing tasks. They are based on principles and may be supported by tools.
- Represent subordinate efforts within projects and are specified by a method and realized by a process.

Because the UML does not prescribe a method or process, it does not specify particular activities.

Heuristics

Heuristics are empirical or experience-based guidelines or rules of thumb. Heuristics

- Facilitate the application of methods as processes.

- Include best practices and lessons learned from experience.
- Represent the knowledge gained from other experiences as well as knowledge gained from previous applications of methods.

The UML encompasses the industry's best engineering practices for problem solving. These practices are realized in the overall organization of the UML and the various diagrams it provides.

,ch02.17200 Page 38 Wednesday, May 9, 2001 4:00 PM