

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

Instructor Notes:

The image shows the cover page of the instructor notes. At the top right is the IBM logo. Below it, centered, is the text 'IBM Software Group'. The main title is 'Essentials of Visual Modeling with UML 2.0' followed by 'Module 2: Principles of Visual Modeling'. Below the title is the 'Rational software' logo. A horizontal decorative bar with various icons is positioned below the logo. The rest of the page is blank.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

Introduce the objectives for this module.

## Objectives

- ◆ Describe the importance of visual modeling and the role of Model Driven Architecture.
- ◆ Define the four principles of visual modeling.
- ◆ Explain what the Unified Modeling Language (UML) represents.
- ◆ Define the type of process that best relates to the UML.

2



# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

Instructor Notes:

Explore the student's  
knowledge of modeling.

## Where Are We?

- ☆ ♦ What is modeling?
- ♦ Four principles of visual modeling
- ♦ The UML
- ♦ Process and visual modeling



3



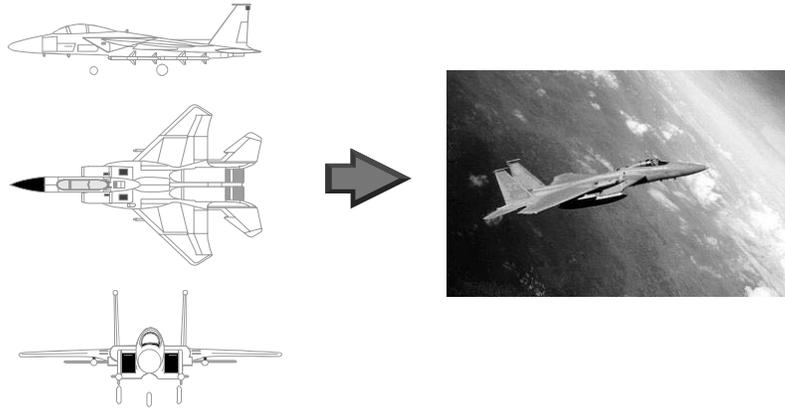
# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Define what is meant by the term "model."

## What Is a Model?

- ◆ A model is a simplification of reality.



According to Grady Booch, IBM Fellow, a model provides the blueprints of a system. Models may encompass detailed plans, as well as more general plans that give a 30,000-foot view of the system under construction. A good model includes those elements that are not relevant to the given level of abstraction. Every system may be described from different aspects using different models, and each model is therefore a semantically closed abstraction of the system. A model may be structural, emphasizing the organization of the system, or it may be behavioral, emphasizing the dynamics of the system.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Sell the students on the value of visual modeling.

Clarify that you are discussing formal modeling, not modeling written on a white board or on the back of a napkin at lunch.

## Why Model?

- ◆ Modeling achieves four aims:
  - Helps you to visualize a system as you want it to be.
  - Permits you to specify the structure or behavior of a system.
  - Gives you a template that guides you in constructing a system.
  - Documents the decisions you have made.
- ◆ You build models of complex systems because you cannot comprehend such a system in its entirety.
- ◆ You build models to better understand the system you are developing.

5



According to Booch in *The Unified Modeling Language User Guide*, modeling achieves four aims:

1. Models help you to **visualize** a system, as you want it to be. A model helps the software team communicate the vision for the system being developed. It is difficult for a software team to have a unified vision of a system that is described only in specification and requirement documents. Models bring about understanding of the system.
2. Models permit you to **specify** the structure of behavior of a system. A model allows how to document system behavior and structure before coding the system.
3. Models give a template that guide you in **constructing** a system. A model is an invaluable tool during construction. It serves as a road map for a developer. Have you experienced a situation where a developer coded incorrect behavior because he or she was confused over the wording in a requirements document? Modeling helps alleviate that situation.
4. Models **document** the decisions you've made. Models are valuable tools in the long term because they give "hard" information on design decisions. You don't need to rely on someone's memory.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

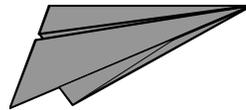
☆ Emphasize that complex projects outside of the software industry require models.

It's inconceivable that a defense contractor would build an airplane as complex as a fighter jet without modeling the airplane first. Why? Because unlike building paper airplanes, the cost of failure is significant.

As a general rule, modeling becomes more important as the complexity and expense rises.

## The Importance of Modeling

Less Important ← → More Important



Paper Airplane



Fighter Jet

6



You can take a piece of paper and a paper clip, and, in a few minutes, have a paper airplane that entertains your kids. If it isn't built just right, you can always start over and build another airplane.

Would it be smart for you to build a fighter jet in the same way? That is, start with some steel, nuts, bolts, and wiring and go right to work. Of course not. You're building an airplane that costs millions of dollars, and the cost of failure is high. You're also be part of a much larger team, needing blueprints and models to effectively communicate with one another. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

- ☆ Point out that a major problem with software development is that we often fail to model complex systems before construction.

Software engineering is an engineering discipline.

Why is it that so many of our projects don't use models like other engineering disciplines such as electrical, civil, and so on?

Would anyone work in a high rise that was put together without the benefit of a well thought-out design?

Would you drive an automobile that had not been modeled first?

Ask the class why they think that software teams don't always model.

Note these ideas and address them through the course.

## Software Teams Often Do Not Model

- ◆ Many software teams build applications approaching the problem like they were building paper airplanes
  - Start coding from project requirements
  - Work longer hours and create more code
  - Lacks any planned architecture
  - Doomed to failure
- ◆ Modeling is a common thread to successful projects

7



If defense contractors want to build fighter jets for the government, they need to achieve a certain balance between the desires of the military with the realities of aerospace engineering. They also want to treat their employees professionally, never placing them at risk or driving them so hard that they burn out.

Curiously, many software development organizations begin wanting to build complex software systems, but approach the problem as though they were building a paper airplane.

With the increasing demand to build more complex software in shorter time, development teams often retreat to the only thing they know how to do well - pound out lines of code. Developers start working longer hours and frequently produce code with a requirements document as their only source of input. However, there eventually comes a time when the application collapses due to the lack of a well thought-out architecture. Consequently, many of these software projects result in failure.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

*This is a good point to introduce students to the DEV325 Essentials of Model Driven Architecture course.*

The MDA framework was adopted by the Object Management Group (OMG) in 2001. Visit [www.omg.org](http://www.omg.org) for additional information.

## Model Driven Architecture (MDA)

- ◆ An approach to using models in software development
  - Separate the specification of the operation of a system from the details of the way that system uses the capabilities of its platform.
    - specifying a system independently of the platform that supports it
    - specifying platforms
    - choosing a particular platform for the system
    - transforming the system specification into one for a particular platform

8



The Model-Driven Architecture prescribes certain kinds of models to be used, how those models may be prepared, and the relationships of these different models.

It's termed *model-driven* because this provides a means for using models to guide the understanding, design, construction, deployment, operation, maintenance and modification.

The *architecture* of a system is the specification of the parts and connectors of the system and the rules for the interactions of the parts using the connectors.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

The *computation independent viewpoint* focuses on the environment of the system, and the requirements for the system; the details of the structure and processing of the system are hidden or as yet undetermined.

The *platform independent viewpoint* focuses on the operation of a system while hiding the details necessary for a particular platform. A platform independent view shows that part of the complete specification that does not change from one platform to another. A platform independent view may use a general purpose modeling language, or a language specific to the area in which the system will be used.

The *platform specific viewpoint* combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system.

## MDA Viewpoints

- ◆ Computational Independent Model (CIM)
  - Focus is on environment of the system and requirements for the system
- ◆ Platform Independent Model (PIM)
  - Focus is on system operation, independent of platform
- ◆ Platform Specific Model (PSM)
  - Focus is on detailed usage of system on specific platform



A *viewpoint* on a system is the process of suppressing selected detail to establish a simplified model, in order to focus on particular concerns within that system.

A CIM does not show details of the structure of systems and is sometimes called a domain model. The vocabulary used in its specification is familiar to the practitioners of the domain in question. The CIM plays an important role in bridging the gap between those that are experts in the domain and its requirements, and those that are experts in the design and construction of the artifacts that together satisfy the requirements.

A PIM exhibits a specified degree of platform independence so as to be suitable for use with a number of different platforms. A very common technique for achieving this independence is to target a system model for a technology-neutral virtual machine. A virtual machine is defined as a set of parts and services (communications, scheduling, naming, etc.), which are defined independently of any specific platform and which are realized in platform-specific ways on different platforms.

A PSM combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

Instructor Notes:

Transition to the next subject.

## Where Are We?

- ◆ What is modeling?
- ☆ ◆ Four principles of visual modeling
- ◆ The UML
- ◆ Process and visual modeling



10



# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

Demonstrate that there are guiding principles for visual modeling. It is not chaotic activity.

The four modeling principles are described in detail on the next four slides.

See the "UML User Guide" for a more detailed discussion on these four principles.

## Four Principles of Modeling

- ◆ The model you create influences how the problem is attacked.
- ◆ Every model may be expressed at different levels of precision.
- ◆ The best models are connected to reality.
- ◆ No single model is sufficient.

11



Modeling has a rich history in all the engineering disciplines.

The four basic principles of modeling are derived from this history.

1. The models you create profoundly influence how a problem is attacked and how a solution is shaped.
2. Every model may be expressed at different levels of precision.
3. The best models are connected to reality.
4. No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Remind students that each model should reflect the problem that they are trying to solve.

The model of a system comprises many different views of that system.

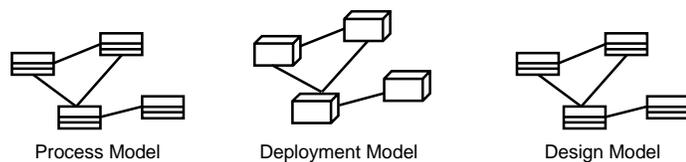
Refer students to the "Architectural Blueprints -The 4+1 View of Software Architecture" by Phillippe Kruchten. You can find the white paper at <http://www-306.ibm.com/software/rational/info/literature/design.jsp>.

Discuss the views later in this module.

Supplement this slide with the following example. To build a home, you would use different models to resolve structural and electrical problems.

## Principle 1: The Choice of Model is Important

- ◆ The models you create profoundly influence how a problem is attacked and how a solution is shaped.
  - In software, the models you choose greatly affect your world view.
  - Each world view leads to a different kind of system.



12



The right models illuminate the most difficult development problems, offering insight that you could not gain otherwise. The wrong models mislead you, causing you to focus on irrelevant issues.

In software, the models you choose can greatly affect your world view. If you build a system through the eyes of a database developer, you'll likely end up with entity-relationship models that push behavior into stored procedures and triggers. If you build a system through the eyes of an object-oriented developer, you'll end up with a system that has its architecture centered around many classes and patterns of interaction that direct how those classes work together.

Each world view leads to a different kind of system with different costs and benefits. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Remember, there are many different ways to view a model of a solution.

Again, point out that there is no one mega-model that displays everything about a system.

Additional examples from building a home may include examples that the homebuyer may be interested only in the elevation and the floor plan.

Obviously, the construction foreman is going to need a more detailed view of the home.

## Principle 2: Levels of Precision May Differ

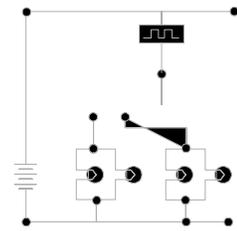
### ◆ Every model may be expressed at different levels of precision.

#### ▪ The best kinds of models let you choose your degree of detail, depending on:

- Who is viewing the model.
- Why they need to view it.



View for Customers



View for Designers

13



If you are building computer chips, sometimes you need a 30,000-foot view. For example, you need your investors to visualize the end product. Other times, you need to get down to the level of the circuits.

When developing a GUI system, a quick and dirty executable model of the user interface may be all you need to communicate your intentions. Other times, when you are dealing with cross-system interfaces of network bottlenecks, you need to model down to the bit level. In either case, the best models are those that let you choose your degree of detail, depending on who is doing the viewing and why they need to view it. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

A model should accurately reflect the real world.

### Principle 3: The Best Models Are Connected to Reality

- ◆ All models simplify reality.
- ◆ A good model reflects potentially fatal characteristics.



14

IBM

A physical model of a building that doesn't respond the same way as the real materials has limited value. It's best to have models that have a clear connection to reality. Where that connection is weak, you need to know exactly how those models are divorced from the real world.

All models simplify reality. The trick is to be sure that your simplifications don't mask any important details. A good model reveals any potentially fatal flaws in design. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

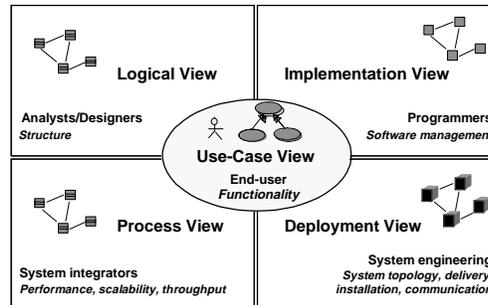
Many interrelated models are needed to accurately describe a system.

Introduce the 4+1 views of architecture first.

It's important that they understand from the beginning that the models created/read represent a specific view of the system.

## Principle 4: No Single Model Is Sufficient

- ◆ No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models.
  - Create models that can be built and studied separately, but are still interrelated.



15



The key phrase is “nearly independent,” meaning that models can be built and studied separately, but are still interrelated.

To understand the architecture of object-oriented systems, you need several complementary and interlocking views. An architectural view can be defined as a simplified description (an abstraction) of a system from a particular perspective or vantage point, covering particular concerns, and omitting entities that are not relevant to this perspective. Views are “slices” of models.

Each of the views below may have structural and behavioral aspects. Together, they represent the blueprints of a software system.

- **Use-case view** exposing the requirements of the system
- **Logical view** capturing the vocabulary of the problem space and the solution space
- **Process view** modeling the distribution of the system’s processes and threads
- **Implementation view** addressing the physical realization of the system
- **Deployment view** focusing on system engineering issues

To address these different needs, Rational has defined the “4+1 view” architecture model.

Remember that not all systems require all views. The number of views is dependent on the system you’re building. For example, a single processor does not require a deployment view or a small program does not require an implementation view and so on.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

Instructor Notes:

## Where Are We?

- ◆ What is modeling?
- ◆ Four principles of visual modeling
- ☆◆ The UML
- ◆ Process and visual modeling



16



# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Introduce the UML.  
Remember many of your students have no idea what the UML really is.

The next four slides explain how the UML is a language for visualizing, specifying, constructing and documenting.

## What Is the UML?

- ◆ The UML is a language for
    - Visualizing
    - Specifying
    - Constructing
    - Documenting
- the artifacts of a software-intensive system.



17



The software systems that you develop today are more complex than the human mind can comprehend. This is why you model systems. Your model selection profoundly influences how you attack the problem and shape the solution.

No single model is sufficient. Every complex system is best approached through a small set of nearly independent models.

Therefore, to increase comprehension, a common language like the Unified Modeling Language (UML) is used to express models.

A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of a system. A modeling language like the UML is a standard language for software blueprints.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

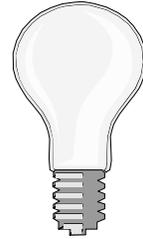
- ☆ Stress how the UML is designed to promote communication using pictures rather than text.

Using the UML allows the “light bulb” to go on in the minds of many people. Rather than trying to interpret a textual description of a system design, the UML offers a graphical representation of that same description.

In this case, a picture is truly worth a thousand words.

## The UML Is a Language for Visualizing

- ◆ Communicating conceptual models to others is prone to error unless everyone involved speaks the same language.
- ◆ There are things about a software system you can't understand unless you build models.
- ◆ An explicit model facilitates communication.



18



Typically, projects and organizations develop their own language for modeling systems, making it difficult for outsiders and new team members to understand what is going on.

Communicating these conceptual models to others is prone to error unless everyone involved speaks the same language. The UML offers a set of symbols that represents well-defined semantics. One developer can write a model in the UML, and another developer can interpret that model unambiguously.

There are things about a software system you can't understand unless you build models that transcend the textual programming language. For example, the meaning of a class hierarchy can be inferred, but not directly grasped, by staring at the code for all the classes in the hierarchy. The UML is a graphical language that addresses this problem.

If the developer who cut the code never wrote down the models, the information would be lost forever. At best, the information would only be partially recoverable from the implementation after the developer has moved on. Writing models in the UML addresses this issue. An explicit model facilitates communication. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

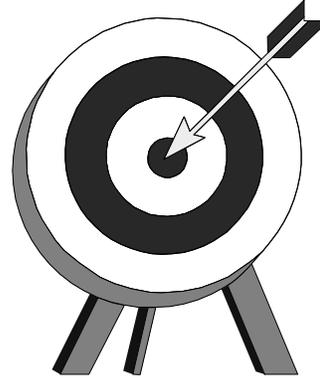
The UML can be used to specify detailed or general models.

Anyone who has worked on a project where miscommunication occurred appreciates this feature of the UML.

The UML allows the modeler to specify their intentions in a clear, unmistakable manner.

## The UML Is a Language for Specifying

- ◆ The UML builds models that are precise, unambiguous, and complete.



19



In this context, **specifying** means to build models that are precise, unambiguous, and complete. In particular, the UML addresses the specification of all the important analysis, design, and implementation decisions that must be made to develop and deploy software-intensive systems. (*The Unified Modeling Language User Guide*, Booch, 1999.)

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

The UML was designed with forward and reverse engineering in mind.

Rational has partners who provide round-trip engineering for other languages.

## The UML Is a Language for Constructing

- ◆ UML models can be directly connected to a variety of programming languages.
  - Maps to Java, C++, Visual Basic, and so on
  - Tables in a RDBMS or persistent store in an OODBMS
  - Permits forward engineering
  - Permits reverse engineering

20



The UML is not a visual programming language. However, models using the UML can be directly connected to a variety of programming languages, making it possible to map from a model in the UML to a programming language or even to a database.

If it is best expressed graphically, it is done graphically in the UML. If it is best expressed textually, it is done in the programming language.

This mapping permits **forward engineering**: the generation of code from a UML model to a programming language. **Reverse engineering** is also possible: the reconstruction of a model from implementation back into the UML.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

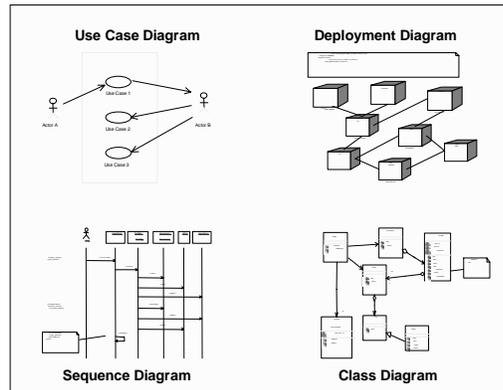
UML diagrams should be treated as formal project artifacts.

Each diagram created by a project team should be treated as an artifact.

The UML can help alleviate some of the paper crunch that many software teams experience.

## The UML Is a Language for Documenting

- ♦ The UML addresses documentation of system architecture, requirements, tests, project planning, and release management.



21



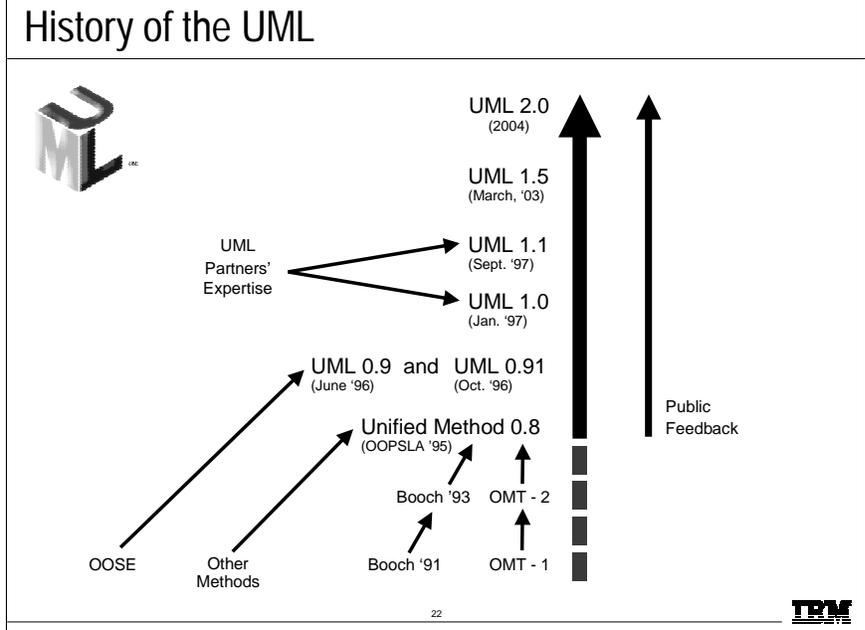
Project artifacts are critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management. (*The Unified Modeling Language User Guide*, Booch, 1999.)

This slide does not represent all the diagrams defined in the UML specification. For example, there is no graphic presented here for a composite structure diagram or a timing diagram. Composite structure is a more advanced modeling concept that is not covered in this course. The timing diagram is new to UML2 and will be introduced in a later module.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:



The UML 2.0 is defined by two complementary specifications, the *Infrastructure* and the *Superstructure*. The UML infrastructure defines foundational concepts that can be used in part or entirely by other specifications. The UML superstructure defines the complete UML. The superstructure specification is self contained and you won't have to read the infrastructure specification unless you are concerned about configuring other specifications in parallel to UML.

The UML metamodel (a description of a model) is divided into two main packages, structure and behavior with two supporting packages, auxiliary elements and profiles.

- The structure package defines the static structure of the UML.
- The behavior package defines the dynamic structure of the UML.

Each package is described by a chapter in the superstructure specification document.

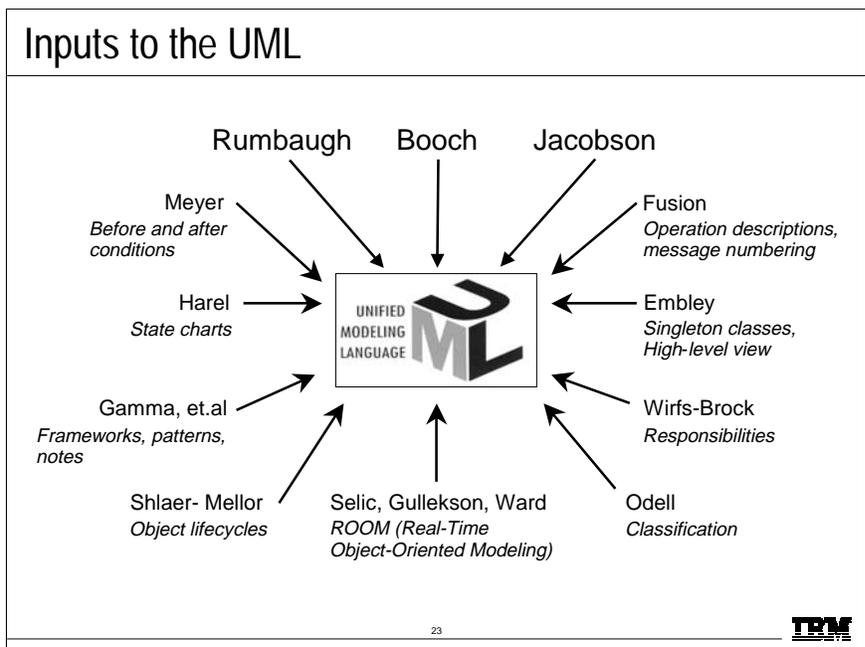
# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Demonstrate that UML was developed as an industry standard with many influences. The UML is not owned and written by Rational.

Do not spend a lot of time on this slide. Simply point out one or two contributors of special interest to your audience.

For example, for a telephony audience, you might point out Harel and his state charts.



UML development included incorporating ideas from numerous other methodologists. The main challenge was to construct an approach that was simple, yet allowed the modeling of a broad range of systems. The conceptual framework was established quickly, but the notational semantics took more time.

Active collaboration with other industry leaders has brought unique expertise and experience into the UML effort. The UML effort was supported by a cross-section of the industry. Partners in the UML effort included HP, ICON Computing, IBM, I-Logix, Intellicorp, MCI Systemhouse, Microsoft, ObjecTime, Oracle, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software, Taskon, and Unisys. These partners provided contributors, reviewers, and advocates for the standardization efforts.

In the end, a modeling language was created that has already withstood the test of widespread use in the industry and the scrutiny of international standardization efforts.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

Instructor Notes:

## Where Are We?

- ◆ What is modeling?
- ◆ Four principles of visual modeling
- ◆ The UML
- ☆ ◆ Process and visual modeling



24



# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Emphasize here that the UML was not designed to stand alone, a process is also needed.

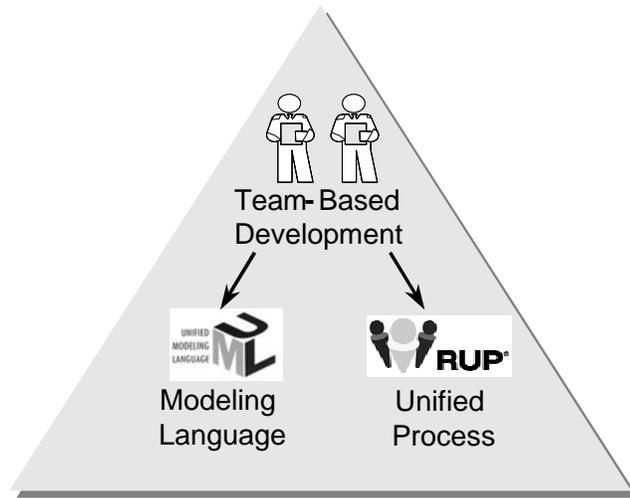
This slide establishes the need for more than the UML.

Some students may think that, if they know the UML, they're done. They don't need a process.

Establish that the UML, while a good and necessary first step, is not enough.

Just knowing the UML would be analogous to learning English by using a dictionary. Sure, all the words are there, but you still need to understand how the English language is structured before one can effectively speak.

## A Language Is Not Enough to Build a System



25



The UML provides a standard for the artifacts of development (semantic models, syntactic notation, and diagrams) that must be controlled and exchanged. But the UML is not a standard for the development process.

Despite all its value, you cannot achieve successful development of today's complex systems solely by using the UML. Successful development also requires employing an equally robust development process.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

---

## Instructor Notes:

UML designers had a specific type of process in mind when the UML was first built.

### What Type of Process Most Benefits the UML?

- ◆ The UML is largely process independent. A process fully benefits from the UML when the process is:
  - Use-case driven
  - Architecture centric
  - Iterative and incremental

26



# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

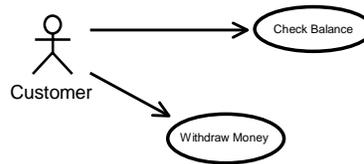
## Instructor Notes:

☆ Use cases are the cornerstone of the type of process that works best with UML. UML diagrams can be used to model a use case.

Use cases are defined later in the course.

## A Use-Case Driven Process

- ◆ Use cases defined for a system are the basis for the entire development process.
- ◆ Benefits of use cases:
  - Concise, simple, and understandable by a wide range of stakeholders.
  - Help synchronize the content of different models.



27



Use cases are one recommended method for organizing your requirements. Instead of a bulleted list of requirements, you organize them in a way that shows how someone can use the system. By doing so, a requirement is more complete and consistent. You can also better understand the importance of a requirement from a user perspective.

It's often difficult to tell how a system does what it is supposed to do from a traditional object-oriented system model. This stems from the lack of a "thread" through the system when it performs certain tasks. Use cases are that thread because they define the behavior performed by a system.

Use cases are not part of "traditional" object orientation, but their importance has become more and more apparent, further emphasized that use cases are part of the UML.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

- ☆ Architecture should drive the design rather than the other way around. UML diagrams allow you to reflect this.

The "DEV475 Mastering Object-Oriented Analysis and Design with UML 2.0" course explores this topic in detail.

## An Architecture-Centric Process

- ◆ A system's architecture is used as a primary artifact for conceptualizing, constructing, managing, and evolving the system under development.
- ◆ Benefits:
  - Intellectual control over a project to manage its complexity and to maintain system integrity.
  - Effective basis for large-scale reuse.
  - A basis for project management.
  - Assistance in component-based development.

28



Use cases drive the process end-to-end over the entire lifecycle. The design activities are centered around architecture-centric architecture, or for software-intensive systems, software architecture. The main focus of the early iterations of an architecture-centric process is to produce and validate a software architecture, which in the initial development cycle takes the form of an executable architectural prototype that gradually evolves to become the final system in later iterations.

A complex system is more than the sum of its parts, more than a succession of small independent tactical decisions. It must have some unifying, coherent structure to organize those parts systematically, and provide precise rules on how to grow the system without having its complexity "explode" beyond human understanding. Architecture provides this structure and these rules.

By clearly articulating the major components and the critical interfaces among them, architecture lets you reason about reuse, both internal reuse (the identification of common parts), and external reuse (the incorporation of ready-made, off-the-shelf components). Architecture can also help reuse on a larger scale. That is, the reuse of the architecture itself in the context of a product line that addresses different functionality in a common domain.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Design the system in small digestible pieces. UML diagrams can enable one to design in an iterative fashion.

Developing iteratively means developing in small pieces that we call use-case realizations.

## An Iterative and Incremental Process

- ◆ Critical risks are resolved before making large investments.
- ◆ Initial iterations enable early user feedback.
- ◆ Testing and integration are continuous.
- ◆ Objective milestones focus on the short term.
- ◆ Progress is measured by assessing implementations.
- ◆ Partial implementations can be deployed.

29



An iterative approach allows users to be involved in a meaningful way throughout the project life cycle. Since each iteration produces an executable release, users can observe the partially executing system and provide meaningful feedback to their level of satisfaction. This ensures that the final system delivered to users is acceptable.

**Continuous testing and integration** ensures that, at every iteration, the pieces all fit together and that the system-level requirements (for example, performance and capacity) are being met.

The **short-term focus of an iteration** is an objective measure of “doneness.” Either a class is included in the build or it’s not. It can’t be 90 percent done. Either a test is executed successfully or it’s not. There is very little room for subjective estimates.

In spite of the best efforts of the development team, not all system features may be complete on the original delivery date. With a waterfall approach, this would usually mean that nothing was ready to be delivered (everything is 90 percent done).

With an iterative approach, most of the system is already fully tested and operational. In many cases, there is value in delivering the partial system on the promised date with the remaining features incorporated at a later time. This can be important when a user has a critical need for some of the new functionality provided by the system or it may help the training and installation process to proceed on schedule. An iterative approach can avoid inconvenient re-planning and re-assignment of resources.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

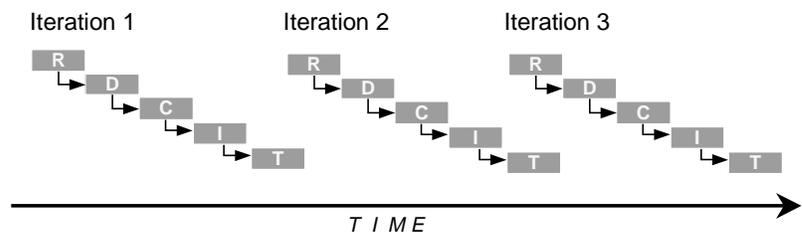
## Instructor Notes:

Define iterative development for the students. It is not necessary to go into great detail at this time. The Mastering Object-Oriented Analysis and Design with UML class discusses this topic.

In the iterative approach, the waterfall is applied to a single increment of the system at a time. Each iteration produces an executable.

Verify that the waterfall concept is understood by the students.

## Iterative Development



- ◆ Earliest iterations address greatest risks.
- ◆ Each iteration produces an executable release, an additional increment of the system.
- ◆ Each iteration includes integration and test.

30



In the diagram above, the following abbreviations are used.

- R Requirements analysis
- D Design
- C Coding, unit testing
- I Implementation
- T Subsystem and system test

Iterative processes were developed in response to these waterfall characteristics. With an iterative process, apply the waterfall steps **iteratively**. Instead of developing the whole system at once, select and develop an increment (a subset of system functionality), then another increment, and so on.

Develop the first increment based on risk, with the highest priority risks to be developed first.

To address the selected risk(s), choose a subset of use cases.

Develop the smallest number of use cases that allow objective verification (like a set of executable tests) of the risks that you've chosen.

Select the next increment to address the next highest risk, and so on.

Thus, you apply the waterfall within each iteration and the system evolves incrementally.

# Essentials of Visual Modeling w/ UML 2.0 - Instructor Notes

## Instructor Notes:

Check to see if the students comprehended the material.

**A.** A model is a simplification of reality.

**B.** The three viewpoints of MDA are: (1) CIM – focus is on environment and requirements; (2) PIM – focus is on system operation, independent of the platform; (3) PSM – focus is on detailed usage on specific platform.

**C.** The four principles of modeling are: (1) The models you create profoundly influence how a problem is attacked and how a solution is shaped; (2) Every model may be expressed at different levels of precision; (3) The best models are connected to reality; (4) No single model is sufficient.

**D.** The UML is a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.

**E.** The UML is largely process independent but consider a process that is: use-case driven, architecture-centric, and iterative and incremental.

**F.** An iteration encompasses the development activities that lead to a product **release** - a stable, executable versions of product, together with any other peripheral elements necessary to use this release.

## Review

- ◆ What is a model?
- ◆ What are the viewpoints of MDA? Describe each one.
- ◆ What are the four principles of modeling? Describe each one.
- ◆ What is the UML? Describe each of its four benefits.
- ◆ What process characteristics best fit the UML? Describe each characteristic.
- ◆ What is an iteration?



31



