

# Introduction to Rational Unified Process

## مقدمة إلى عقلاني موحد عملية



Modified in many cases to support instructional needs. Original developed by Rational

# What Is a Process?

## ما هي العملية؟

A process defines **Who** is doing **What**, **When** and **How** to reach a certain goal. In software engineering the goal is to build a software product or to enhance an existing one

تحدد العملية من يفعل ماذا ومتى وكيف يصل إلى هدف معين. الهدف في هندسة البرمجيات هو بناء منتج برمجي أو تحسين منتج موجود



العمال:	Workers:	the <b>WHO</b>	من
الأنشطة:	Activities:	the <b>HOW</b>	كيف
المشغولات:	Artifacts:	the <b>WHAT</b>	ماذا
سير العمل:	Workflows:	the <b>WHEN</b>	متى

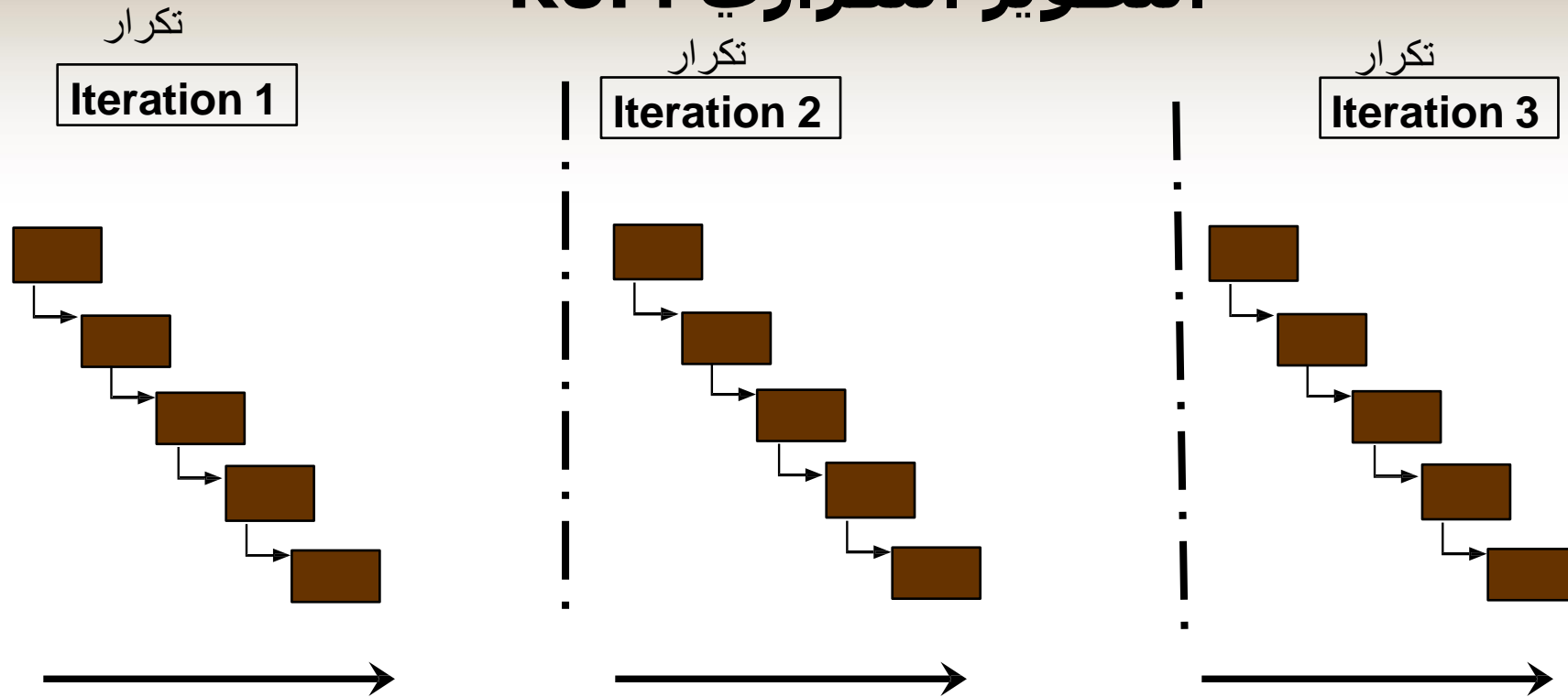
# RUP Characteristics

## RUP خصائص

- ▶ Rational Unified Process is best described as an  
أفضل وصف للعملية الموحدة العقلانية بأنها ▶
  - Iterative
    - ترابطي
  - Use-case driven
    - استخدام حالة مدفوعة
  - Architecture-centric
    - تتمحور العمارة

# RUP: Iterative Development

## التطوير التكراري RUP



- ➔ Earliest iterations address greatest risks

- Each iteration produces an executable release

- Each iteration includes integration, test, and assessment!

- Objective Milestones: short-term focus; short term successes!

- التكرارات المبكرة تعالج أكبر المخاطر

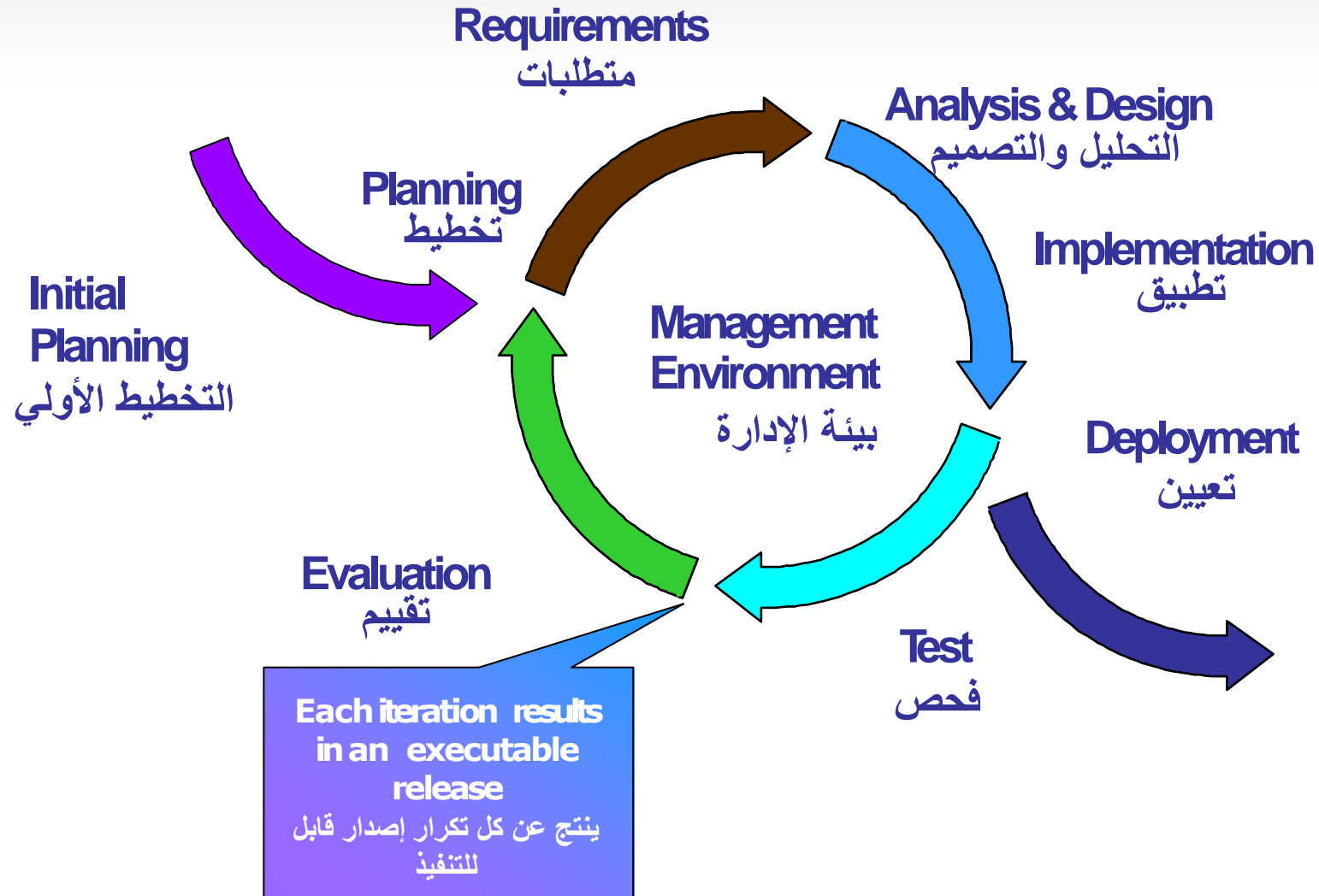
- ينتج عن كل تكرار إصدار قابل للتنفيذ

- يتضمن كل تكرار التكامل والاختبار والتقييم!

- المعالم الموضوعية: تركيز قصير المدى ؛ نجاحات قصيرة المدى!

# Iterative Process Produces a Release

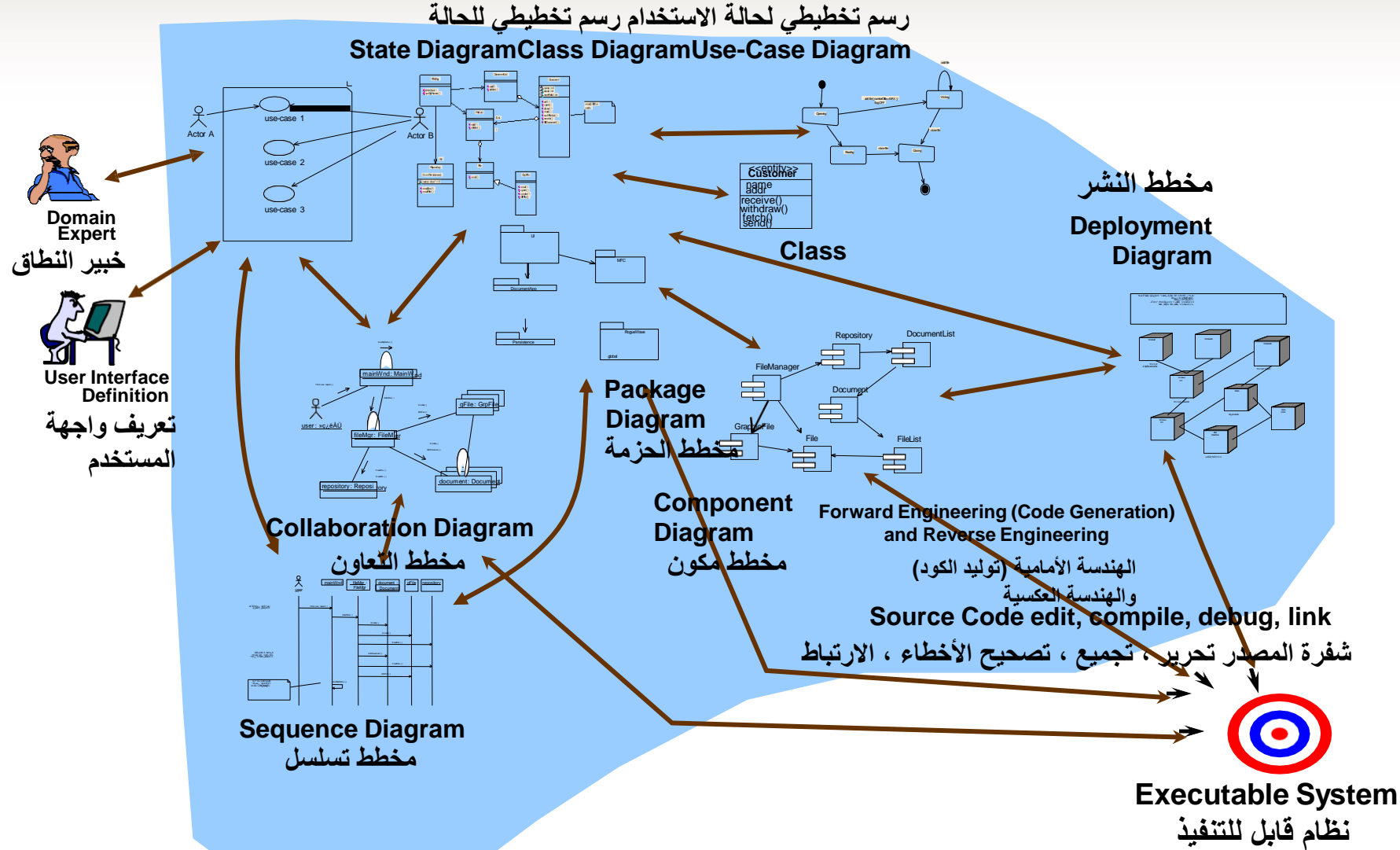
## ينتج عن العملية التكرارية إصدار



# RUP: Use-case driven RUP: حالة الاستخدام مدفوعة

- A use-case driven process means that the use-cases designed for a system are the basis for the entire development process:

► تعني العملية المدفوعة بحالة الاستخدام أن حالات الاستخدام المصممة لنظام ما هي الأساس لعملية التطوير بأكملها:



# RUP: An Architecture-Centric Process:

## عملية تتمحور حول الهندسة المعمارية:

- Architecture is a primary focus of the early iterations

العمارة هي التركيز الأساسي للتكرارات المبكرة

- Building, validating, and baselining the architecture constitute the **primary objectives** (but not all) of Elaboration Phase – especially the first iteration...

■ يشكل البناء والتحقق من الصحة والبطانة الأساسية للبنية الأهداف الأساسية (ولكن ليس كلها) لمرحلة التفصيل - خاصة التكرار الأول ...

- Benefits of an Architecture-Centric Process

فوائد عملية تتمحور حول العمارة

- (Think 'parts': layers, subsystems, packages, relationships, components, etc....)

■ (فكر في "الأجزاء": الطبقات ، والأنظمة الفرعية ، والحزم ، والعلاقات ، والمكونات ، وما إلى ذلك ...)

- Manage its complexity إدارة تعقيدها
- Provides an effective basis for large-scale reuse
- يوفر أساسا فعالا لإعادة الاستخدام على نطاق واسع
- Provides a basis for project management – allocation to teams
- يوفر أساسا لإدارة المشاريع - تخصيص الموارد للأفرقة

# RUP Structure

## هيكل RUP

The Rational Unified Process is structured along two dimensions:

تم تنظيم العملية الموحدة الرشيدة على أساس بعدين:

- ▶ **Time (Dynamic Structure)** — division of the life cycle into phases and iterations

◀ الوقت (الهيكل الديناميكي) - تقسيم دورة الحياة إلى مراحل وتكرار

- ▶ **Process components (Static Structure)**—production of a specific set of artifacts with well-defined activities

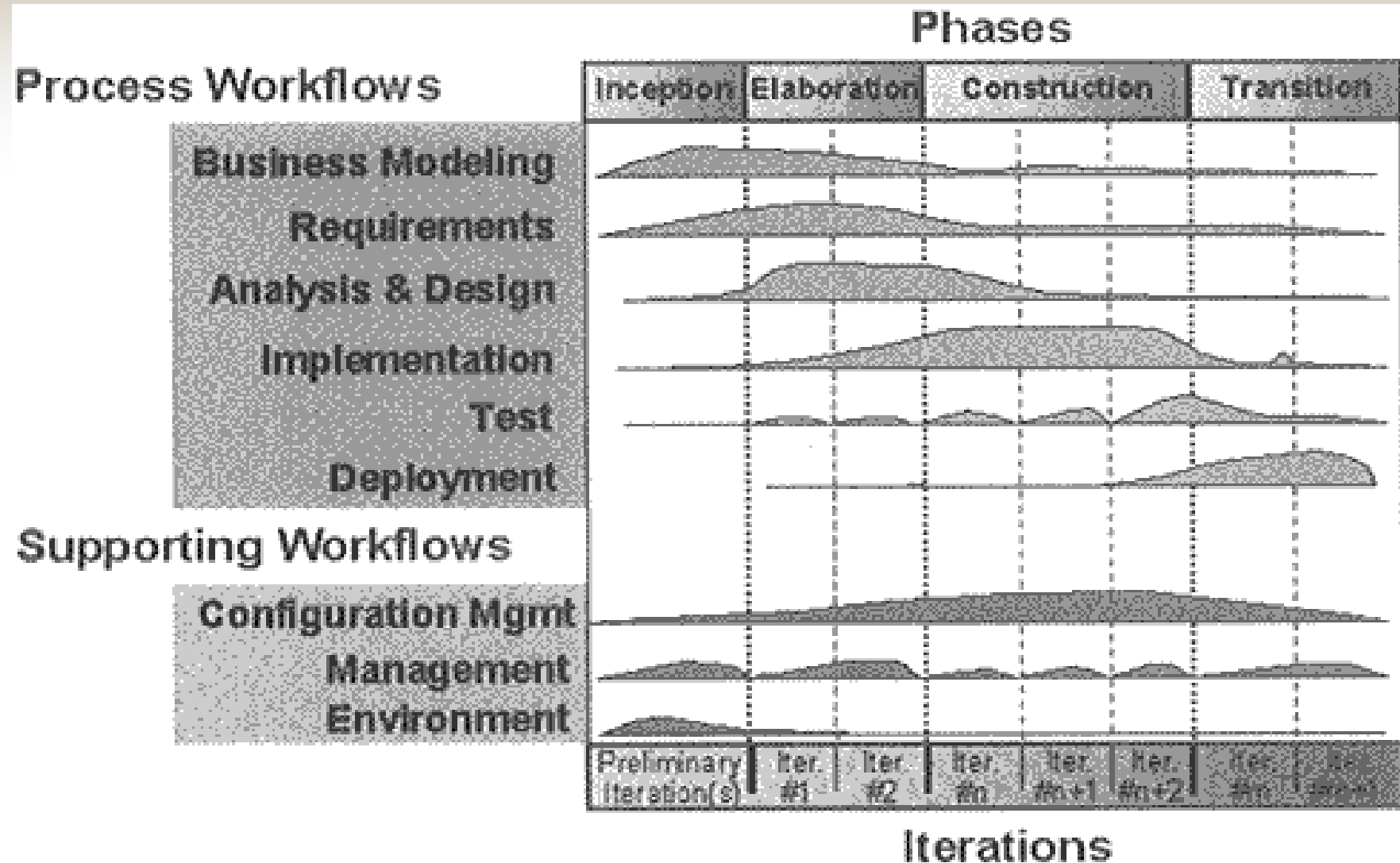
◀ مكونات العملية (الهيكل الثابت) - إنتاج مجموعة محددة من القطع الأثرية ذات أنشطة محددة جيداً

Both dimensions must be taken into account for a project to succeed.

ويجب أن يؤخذ كلا البعدين في الاعتبار لكي ينجح المشروع.



**Process components are applied to each time- based phase**  
**تطبق عناصر العملية على كل مرحلة زمنية**



# Structuring a project along the time dimension involves the adoption of the following time-based phases:

وتشمل هيكله مشروع على طول البعد الزمني اعتماد المراحل الزمنية التالية:



## ▪ **Inception** - Define the scope of project

▪ البدء - تحديد نطاق المشروع

- What is included; what is not; identify all actors and use cases; draft about 20% of essential use cases; High level identification.

► ما هو مشمول ؛ ما ليس كذلك ؛ وتحديد جميع الجهات الفاعلة وحالات الاستخدام ؛ وصياغة نحو ٢٠ في المائة من حالات الاستخدامات الضرورية ؛ تحديد رفيع المستوى.

- Lots of artifacts: Business Rules, Vision, desired Features, Domain Model, costs, schedule, Risk List, Business modeling, and more)

► الكثير من القطع الأثرية: قواعد الأعمال، الرؤية، الميزات المرغوبة، نموذج المجال، التكاليف، الجدول الزمني، قائمة المخاطر، نمذجة الأعمال، والمزيد)

## ▪ **Elaboration** - Plan project, Specify features, use cases are detailed and architectural decisions are made, have about 80% of essential requirements identified; (for us, use-cases)

▪ وضع - مشروع الخطة، وتحديد الخصائص، وتفاصيل حالات الاستخدام، واتخاذ القرارات المعمارية، وتحديد حوالي ٨٠ في المائة من المتطلبات الأساسية ؛ (بالنسبة لنا، حالات الاستخدام)

- Good grasp of requirements and architecture..الهندسة المعمارية..فهم جيد للمتطلبات

- Most key analysis and design done. (often a couple of iterations...)

► تم إجراء معظم التحليلات والتصميمات الرئيسية. (غالبًا ما يكون هناك تكرار...)

## ▪ **Construction** - Construction is where the bulk of the coding is done. Build the product via several iterations; up to beta release; Essentially programming and unit test; iterations; assessment, ... Several iterations (time-boxed)...

▪ البناء - البناء هو المكان الذي يتم فيه الجزء الأكبر من الترميز. بناء المنتج من خلال عدة تكرارات ؛ حتى إطلاق الإصدار التجريبي ؛ البرمجة أساسا واختبار الوحدة ؛ والتكرار ؛ تقييم، ... عدة تكرارات (محاصر زمنية)...

## ▪ **Transition** - Transition the product to end user community; end-user training and support; installation, etc.

▪ الانتقال - نقل المنتج إلى مجتمع المستخدمين النهائيين ؛ وتدريب المستعمل النهائي ودعمه ؛ والتركيب، إلخ.

# عنصر Process Component (Workflows) Dimension العملية (تدفقات العمل)

Structuring the project along the process component dimension includes the following

تشمل هيكلية المشروع على طول بُعد عنصر العملية ما يلي:

- ▶ **Business Modeling** —the study of an organization so we can identify the desired system capabilities and user needs

◀ نمذجة الأعمال - دراسة منظمة حتى تتمكن من تحديد قدرات النظام المرغوب واحتياجات المستخدم

- ▶ **Requirements**—a narration of the system vision along with a set of functional and nonfunctional requirements

◀ المتطلبات - سرد لرؤية النظام إلى جانب مجموعة من المتطلبات الوظيفية وغير الوظيفية

- ▶ **Analysis and Design** —a description of how the system will be realized in the implementation phase

◀ التحليل والتصميم - وصف لكيفية تحقيق النظام في مرحلة التنفيذ

- ▶ **Implementation** —the production of the code that will result in an executable system

◀ التنفيذ - إنتاج المدونة التي ستؤدي إلى نظام قابل للتنفيذ

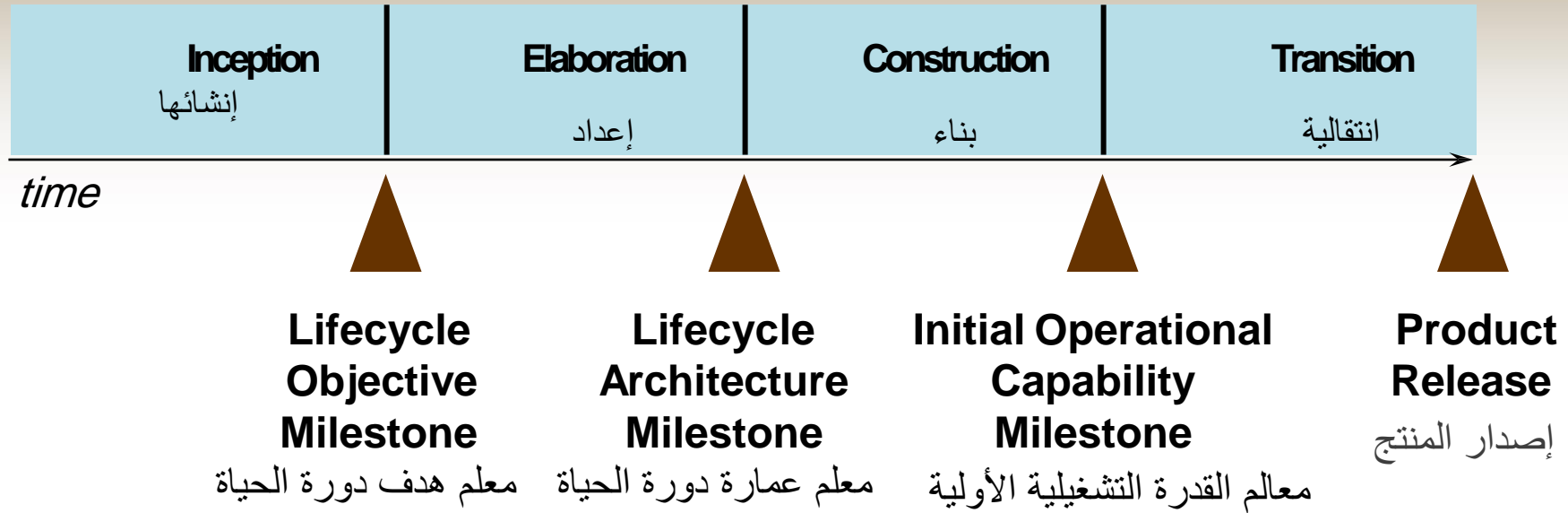
- ▶ **Test** —the verification of the entire system الاختبار - التحقق من النظام بأكمله

- ▶ **Deployment** —the delivery of the system and user training to the customer activities:

◀ النشر - إيصال النظام وتدريب المستخدمين إلى أنشطة العملاء:

# Phase Boundaries Mark Major Milestones

## حدد حدود المرحلة المعالم الرئيسية



Note the titles of the Milestones  
لاحظ عناوين المعالم.

**At each of the major milestones**, the project is reviewed and a decision made as to whether to proceed with the project as planned, to abort the project, or to revise it.

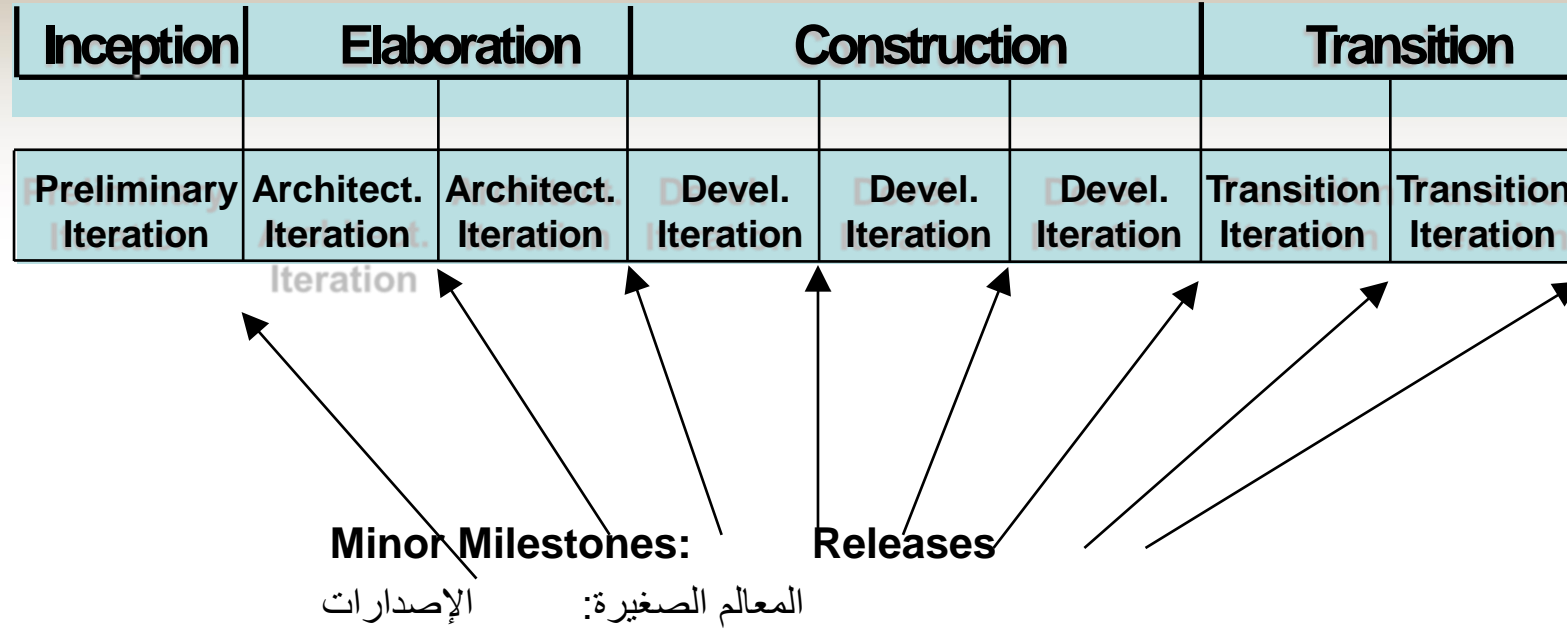
**في كل من المعالم الرئيسية**، يتم استعراض المشروع واتخاذ قرار بشأن ما إذا كان سيتم المضي قدماً في المشروع كما هو مخطط له، أو لإجهاض المشروع، أو تنقيحه.

**These are indeed MAJOR milestones!**

**هذه بالفعل معالم رئيسية!**

# Iterations and Phases – Closer Look at Process Architecture

## التكرارات والمراحل - نظرة فاحصة على بنية العملية



- ➔ An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an 'executable' release (internal or external) and assessment of the iteration

• التكرار هو تسلسل متميز للأنشطة مع خطة ومعايير تقييم راسخة، مما يؤدي إلى إطلاق «قابل للتنفيذ» (داخلي أو خارجي) وتقييم التكرار

- Number of iterations per phase is variable.

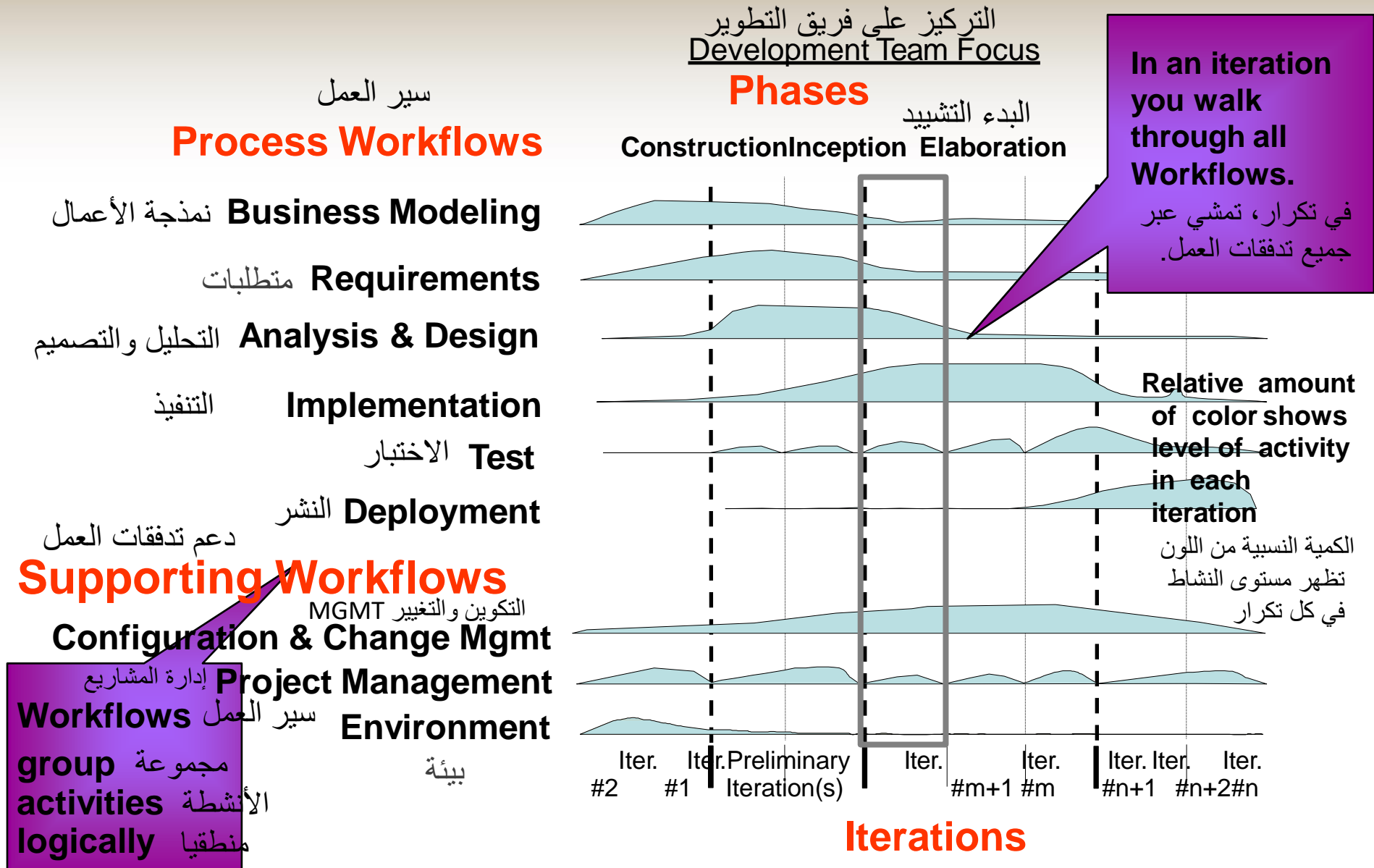
• عدد التكرارات لكل مرحلة متغير.

- Each iteration can result in an executable release where the system evolves into a large system.

• عدد التكرارات لكل مرحلة متغير.

# Bringing It All Together: The Iterative Model

## الجمع بين كل ذلك: النموذج التكراري



# Example

- For example, a two-year project would have the following:

► فعلى سبيل المثال، سيكون لمشروع مدته سنتان ما يلي:

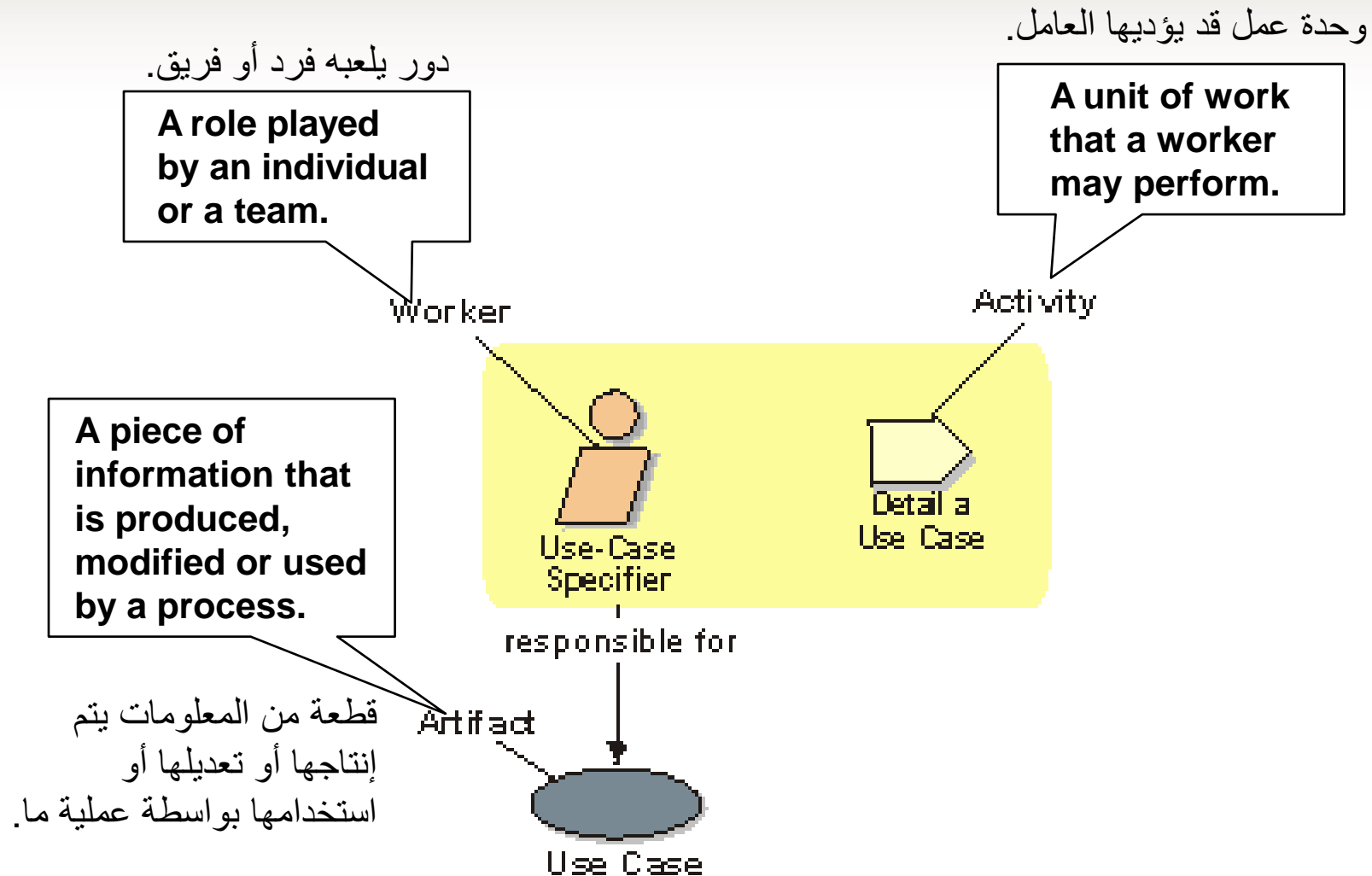
- A 2 ½ month inception phase      مرحلة بدء ٢ ½ شهر
- A 7 month elaboration phase      مرحلة إعداد مدتها ٧ أشهر
- A 12 month construction phase      مرحلة بناء مدتها ١٢ شهرا
- A 2 ½ month transition phase      مرحلة انتقالية ٢ ½ شهر

**Phases length varies greatly depending on the specific circumstances of the project. What is important is the goal of each phase and the milestone that concludes it.**

ويختلف طول المراحل اختلافا كبيرا تبعا للظروف المحددة للمشروع. المهم هو هدف كل مرحلة والمعلم الذي يختتمها.

# Process Notation – Figures used in the RUP

## تدوين العمليات - الأرقام المستخدمة في الحزب الثوري المتحد





# Key Concept: Workflow

## المفهوم الرئيسي: سير العمل

- Sequence of activities that produce a result of observable value

► تسلسل الأنشطة التي تنتج نتيجة لقيمة يمكن ملاحظتها

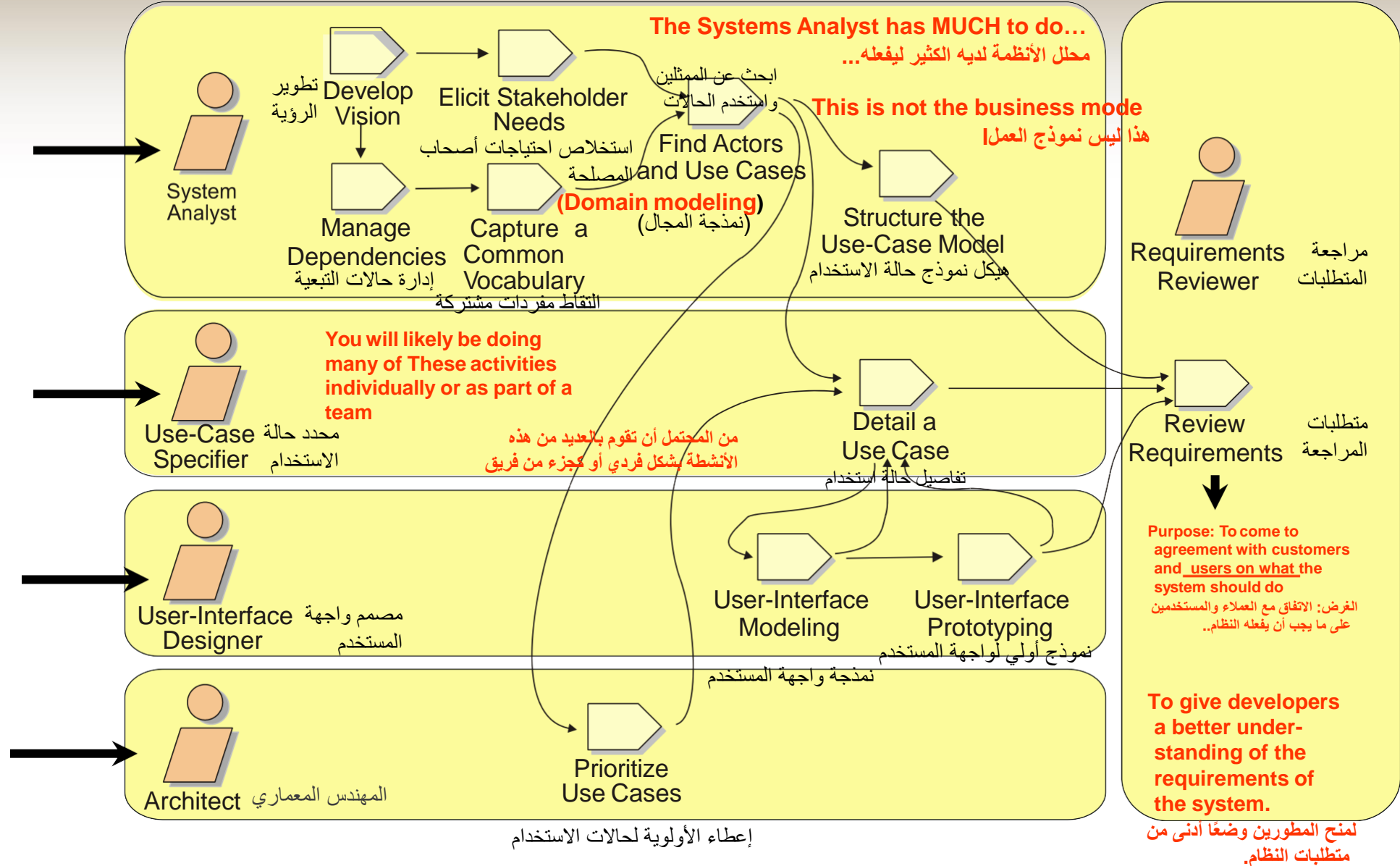
One of  
nine core  
workflows

واحد من تسعة تدفقات  
عمل أساسية



# Requirements Workflow

## متطلبات سير العمل



## Summary: Rational Unified Process

### ملخص: عملية موحدة عقلانية

- ▶ A software development process defines **Who** is doing **What**, **When** and **How** in building a software product
  - ▶ تحدد عملية تطوير البرمجيات من يفعل ماذا ومتى وكيف في بناء منتج برمجي
- ▶ The Rational Unified Process has four phases: **Inception, Elaboration, Construction and Transition**
  - ▶ تتكون العملية الموحدة الرشيدة من أربع مراحل: البدء والتطوير والبناء والانتقال
- ▶ Each phase ends at a major milestone and contains one or more iterations
  - ▶ تنتهي كل مرحلة عند معلم رئيسي وتحتوي على تكرار واحد أو أكثر
- ▶ An **iteration** is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release
  - ▶ التكرار هو تسلسل متميز للأنشطة مع خطة ثابتة ومعايير تقييم، مما يؤدي إلى إصدار قابل للتنفيذ

# Summary (cont.): Rational Unified Process

## ملخص (تابع): عملية موحدة عقلانية

- ▶ A **workflows** groups related activities together

▶ مجموعة من الأنشطة المتصلة بسير العمل

- ▶ Each **workflow** is exercised (to a greater or lesser degree...) during an **iteration** and results in a model that is incrementally produced

▶ يتم ممارسة كل سير عمل (بدرجة أكبر أو أقل...) أثناء التكرار وينتج عنه نموذج يتم إنتاجه بشكل تدريجي

- ▶ An **artifact** is a piece of information that is produced, modified, or used by a process

▶ القطعة الأثرية هي قطعة من المعلومات يتم إنتاجها أو تعديلها أو استخدامها بواسطة عملية

- ▶ A **worker** is a **role** that may be played by an individual or a team in the development organization

▶ العامل هو الدور الذي يمكن أن يقوم به فرد أو فريق في منظمة التنمية

- ▶ An **activity** is a unit of work a worker may be asked to perform

▶ النشاط هو وحدة عمل قد يُطلب من العامل القيام بها

# Software Engineering and Best Practices

Sources: Various.

Rational Software Corporation slides,  
OOSE textbook slides, Per Kroll talk, How to Fail with  
the RUP article, textbooks

Most slides have been modified considerably

# What is software engineering??

## ما هي هندسة البرمجيات؟؟

Software Engineering is an engineering discipline which is concerned with all aspect of software production from the early stages of system specification through to maintaining the system after it has gone into use.

هندسة البرمجيات هي تخصص هندسي يهتم بجميع جوانب إنتاج البرامج من المراحل الأولى لمواصفات النظام وحتى صيانة النظام بعد استخدامه.

# What is software engineering??

## ما هي هندسة البرمجيات؟؟

Software Engineers must adopt a systematic and organized approach to their work and use appropriate tools and techniques depending on the problem to be solved ,development constraints, and the resource availability”

يجب على مهندسي البرمجيات اعتماد نهج منظم ومنظم لعملهم واستخدام الأدوات والتقنيات المناسبة اعتمادًا على المشكلة المراد حلها وقيود التطوير وتوافر الموارد "

# What is a software process ??

## ما هي عملية البرمجيات ؟؟

- ▶ IEEE - A process is a sequence of steps performed for a given purpose
  - ▶ IEEE-العملية عبارة عن سلسلة من الخطوات يتم تنفيذها لغرض معين
- ▶ A set of activities whose goal is the development or evolution of software
  - ▶ مجموعة من الأنشطة التي تهدف إلى تطوير أو تطوير البرمجيات
- ▶ General activities in all software processes :
  - ▶ الأنشطة العامة في جميع عمليات البرمجيات:
    - Specification: what the system should do and it's development constrains
      - المواصفات: ما يجب أن يفعله النظام وقيود التطوير
    - Development: production of the software
      - التطوير: إنتاج البرنامج
    - Validation: checking that the software is what the customer wants
      - التحقق من الصحة: التحقق من أن البرنامج هو ملف
    - Evolution : changing the software in response to changing demands
      - التطور: تغيير البرنامج استجابة للطلبات المتغيرة



# Comments:

## تعليقات:

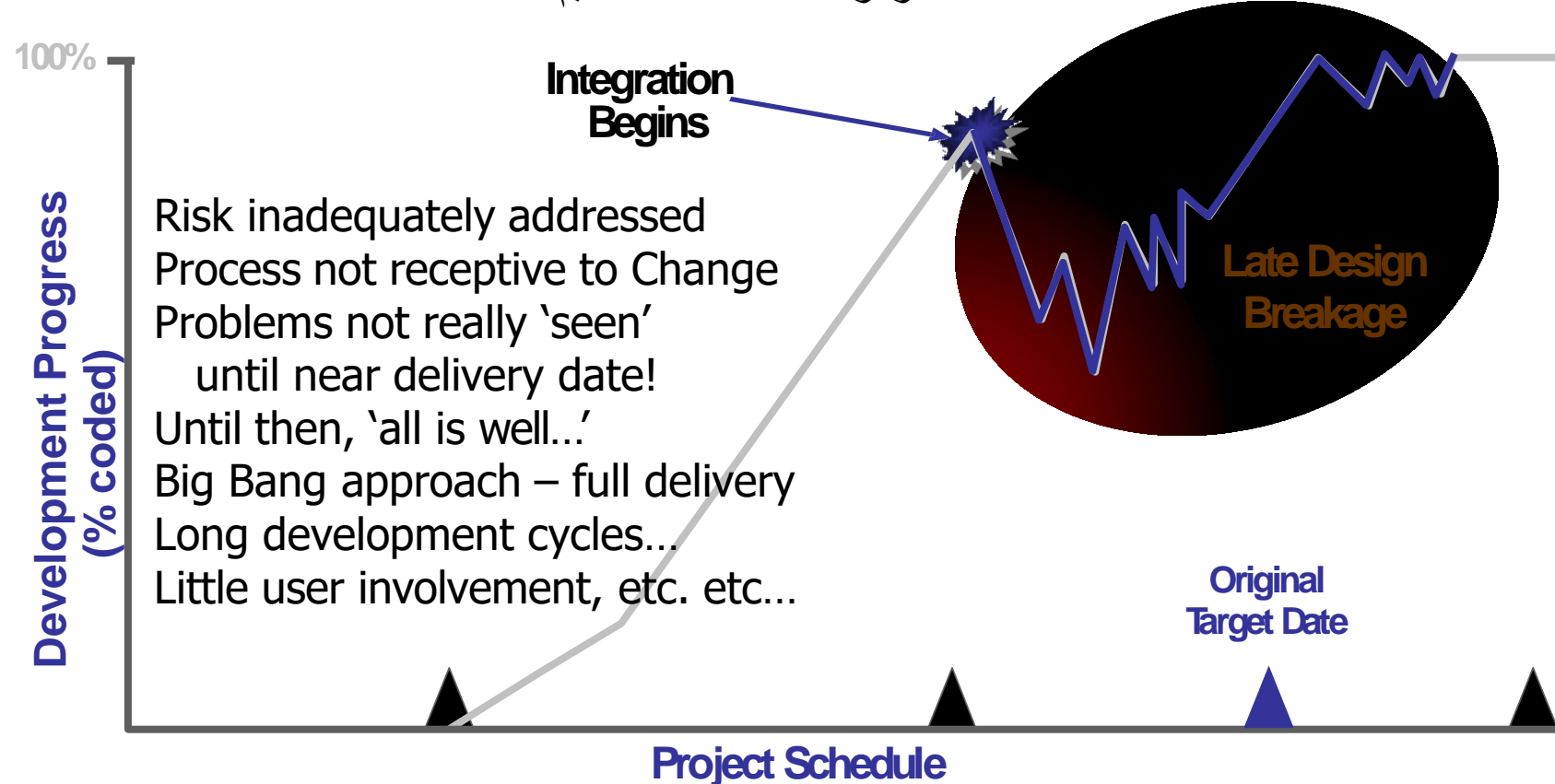
- ▶ \$250 billion annually in US.. ٢٥٠ مليار دولار سنويا في الولايات المتحدة
- ▶ Over 175,000 projects!! أكثر من ١٧٥٠٠٠ مشروع
- ▶ Complexity, size, distribution, importance push our limits. التعقيد والحجم والتوزيع والأهمية تدفع حدودنا.
- ▶ Business pushes these limits: يدفع العمل هذه الحدود:
  - Great demands for rapid development and deployment مطالب كبيرة للتطوير والنشر السريع
- ▶ ➔ Incredible pressure: develop systems that are: ← ضغط لا يُصدق: طور أنظمة هي:
  - On time, في الوقت المحدد،
  - Within budget, في حدود الميزانية ،
  - Meets the users' requirements يفي بمتطلبات المستخدمين
- ▶ Figures in the late 90s indicated that at most تشير الأرقام في أواخر التسعينيات إلى ذلك على الأكثر
  - 70% of projects completed اكتملت ٧٠٪ من المشاريع
  - Over 50% ran over twice the intended budget أكثر من ٥٠٪ تجاوزت ضعف الميزانية المقصودة
  - \$81 billion dollars spent in cancelled projects!! ٨١ مليار دولار صرفت على مشاريع ملغاة !!
- ▶ Getting better, but we need better tools and techniques!! نتحسن ، لكننا بحاجة إلى أدوات وتقنيات أفضل

# What Happens in Practice

## ماذا يحدث في الممارسة العملية

Sequential activities: (Traditional 'Waterfall' Process)

الأنشطة المتسلسلة: (عملية "الشلال" التقليدية)



# The Result is Often Referred to as the Software Crisis

## غالبًا ما يشار إلى النتيجة باسم أزمة البرامج

- ▶ Schedule Overruns تجاوزات الجدول
- ▶ Cost Estimate Overruns تجاوزات تقدير التكلفة
- ▶ Software Quality Problems مشاكل جودة البرمجيات
- ▶ Software does not meet User Expectations
- ▶ البرنامج لا يلبي توقعات المستخدم
- ▶ Productivity of Software Developers has not been keeping up with demand
- ▶ لم تكن إنتاجية مطوري البرامج مواكبة للطلب
  - Especially since the rise of the Internet
  - خاصة منذ ظهور الإنترنت
  - until the internet bubble exploded in the Spring of 2000
  - حتى انفجرت فقاعة الإنترنت في ربيع عام ٢٠٠٠

# Why is This?

## لماذا هذا؟

- ▶ Software is Dynamic versus Static البرنامج ديناميكي مقابل ثابت
- ▶ Software is Complex البرنامج معقد
- ▶ Software is difficult to Conceptualize البرمجيات يصعب تصورها
- ▶ Software is difficult to Represent البرمجيات صعبة التمثيل
- ▶ Software is difficult to Communicate البرمجيات صعبة التواصل
- ▶ Software is difficult to Evaluate and Measure
- ▶ البرامج صعبة التقييم والقياس
- ▶ Software Developers have trouble learning what users want:
  - ▶ يواجه مطورو البرامج صعوبة في معرفة ما يريده المستخدمون:
    - Users do not know what they want المستخدمون لا يعرفون ماذا يريدون
    - Software developers misunderstand the problem
    - يسيء مطورو البرمجيات فهم المشكلة
- ▶ There is a tremendous demand for software
- ▶ هناك طلب هائل على البرامج

# What can we do about it?

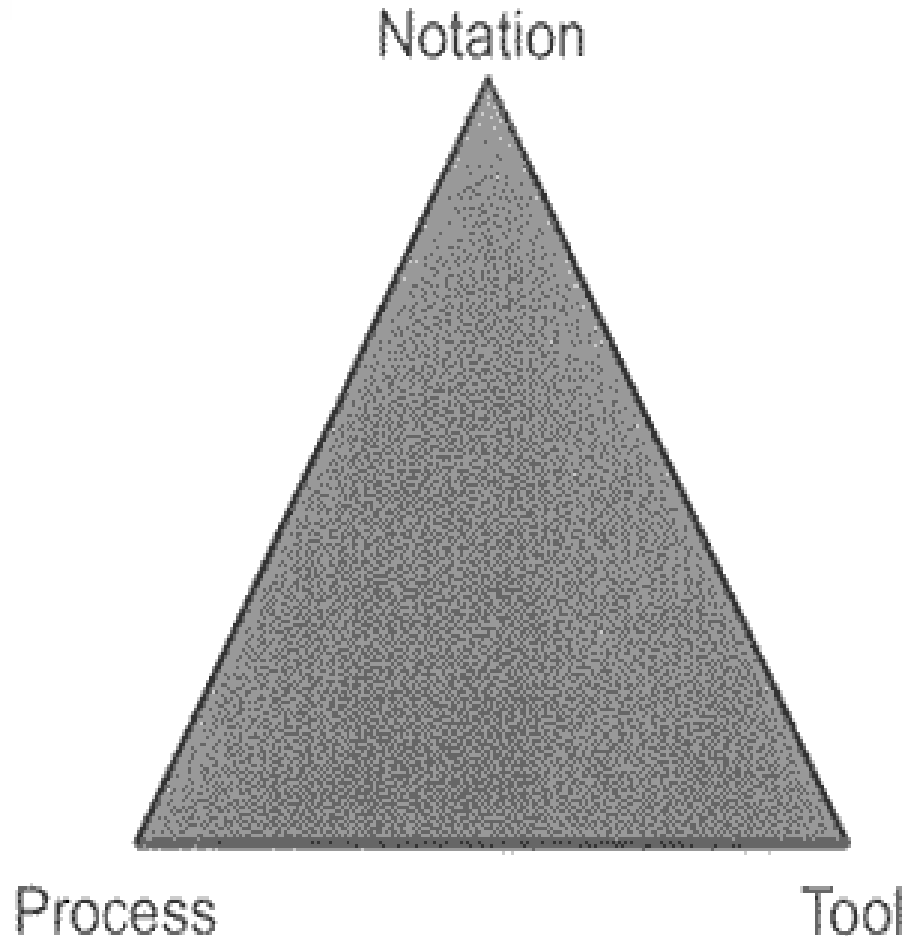
## The Triangle For Success

# مالذي يمكننا فعله حيال هذا؟

## المثلث للنجاح

### The needed elements for a successful project

العناصر اللازمة لمشروع ناجح



- ▶ You need all three facets—a notation, a process, and a tool.
- ▶ أنت بحاجة إلى جميع الأوجه الثلاثة - تدوين وعملية وأداة.
- ▶ You can learn a notation, but if you don't know how to use it (process), you will probably fail.
- ▶ يمكنك تعلم الترميز ، ولكن إذا كنت لا تعرف كيفية استخدامه (العملية) ، فمن المحتمل أن تفشل.
- ▶ You may have a great process, but if you can't communicate the process (notation), you will probably fail.
- ▶ قد تكون لديك عملية رائعة ، ولكن إذا لم تتمكن من توصيل العملية (الترميز) ، فمن المحتمل أن تفشل.
- ▶ And lastly, if you cannot document the artifacts of your work (tool), you will probably fail.
- ▶ وأخيرًا ، إذا لم تتمكن من توثيق القطع الأثرية لعملك (الأداة) ، فمن المحتمل أن تفشل.

# The Role of Process

## دور العملية

- ▶ We need a process that نحن بحاجة إلى عملية
  - Will serve as a framework for large scale and small projects
  - سيكون بمثابة إطار عمل للمشاريع الكبيرة والصغيرة
  - ← التكيف - يحتضن "التغيير!" → Adaptive – embraces 'change!'
    - ▶ Opportunity for improvement not identification of failure! فرصة للتحسين وليس تحديد الفشل!
  - Iterative (small, incremental 'deliverables') تكرارية ("مخرجات" صغيرة ، تدريجية)
  - Risk-driven (identify / resolve risks up front) مدفوعة بالمخاطر (تحديد / حل المخاطر مقدمًا)
  - Flexible, customizable process (not a burden; adaptive to projects)
  - عملية مرنة وقابلة للتخصيص (ليست عبئًا ؛ قابلة للتكيف مع المشاريع)
  - Architecture-centric (breaks components into 'layers' or common areas of responsibility...)
    - تتمحور حول العمارة (يقسم المكونات إلى "طبقات" أو مجالات مسؤولية مشتركة ...)
  - **Heavy** user involvement مشاركة كبيرة من المستخدم
  - Captures and institutionalizes **best practices** of Software Engineering
  - يلتقط ويؤسس أفضل ممارسات هندسة البرمجيات
- ▶ Identify best ways of doing things حدد أفضل الطرق لعمل الأشياء

# The Role of Notation

## دور التدوين

- ▶ Notation plays an important part in any model—it is the glue that holds the process together. ? Notation has three roles:
  - ▶ يلعب التدوين دورًا مهمًا في أي نموذج - إنه الغراء الذي يربط العملية معًا. ؟ للتدوين ثلاثة أدوار:
- ▶ It serves as the language for communicating decisions that are not obvious or can be inferred from the code itself.
  - ▶ إنها بمثابة لغة لتوصيل القرارات غير الواضحة أو التي يمكن استنتاجها من الكود نفسه.
- ▶ It provides semantics that are rich enough to capture all important strategic and tactical decisions.
  - ▶ يوفر دلالات غنية بما يكفي لالتقاط جميع القرارات الاستراتيجية والتكتيكية المهمة.
- ▶ It offers a form concrete enough for humans to reason and for tools to manipulate?
  - ▶ إنه يقدم شكلًا ملموسًا بما يكفي للإنسان للتفكير ولأدوات للتلاعب؟

# The Role of Tool

## دور الأداة

- ▶ Any software development method is best supported by a tool  
يتم دعم أي طريقة لتطوير البرامج بشكل أفضل بواسطة أداة
- ▶ The Rational Rose product family is designed to provide the software developer with a complete set of visual modeling tools for development of robust, efficient solutions to real business needs in the client/server, distributed enterprise, and real-time systems environments.  
تم تصميم عائلة منتجات Rational Rose لتزويد مطور البرامج بمجموعة كاملة من أدوات النمذجة المرئية لتطوير حلول قوية وفعالة لاحتياجات العمل الحقيقية في العميل / الخادم ، والمؤسسات الموزعة ، وبيئات أنظمة الوقت الفعلي.
- ▶ Rational Rose products share a common universal standard, making modeling accessible to non programmers wanting to model business processes as well as to programmers modeling applications logic.  
تتشارك منتجات Rational Rose في معيار عالمي مشترك ، مما يجعل النمذجة في متناول غير المبرمجين الراغبين في نمذجة العمليات التجارية وكذلك للمبرمجين الذين يصممون منطق التطبيقات.



# What is CASE Tool

## ما هي أداة CASE

- ▶ Computer Aided Software Engineering
- ▶ هندسة البرمجيات بمساعدة الحاسوب
- ▶ Software systems which are intended to provide automated support for software process activities
- ▶ أنظمة البرامج التي تهدف إلى توفير الدعم الآلي لأنشطة عمليات البرامج
- ▶ CASE tools are:
- ▶ أدوات CASE هي:
  - 1. Upper-CASE : tools to support early process activities of requirement and design
  - Upper-CASE: أدوات لدعم أنشطة العملية المبكرة للمتطلبات والتصميم
  - 2. Lower-CASE : tools to support later activities such as programming, debugging and testing
  - Low-CASE: أدوات لدعم الأنشطة اللاحقة مثل البرمجة وتصحيح الأخطاء والاختبار

# Why Rational Software??

## لماذا البرامج العقلانية؟؟

- ▶ Rational software covers the entire lifecycle of software production.  
تغطي البرامج العقلانية دورة الحياة الكاملة لإنتاج البرامج.
- ▶ Rational software provides methodology (process) and tools to cover all aspects of software production.  
توفر البرمجيات العقلانية منهجية (عملية) وأدوات لتغطية جميع جوانب إنتاج البرمجيات.

# Unit 3 : Use Cases and Actors

## Unit 3 Theoretical

Source: Visual Modeling with Rational Rose 2002

Khalil Barhoum, Isa University

Email: [kbarhoum@ipu.edu.jo](mailto:kbarhoum@ipu.edu.jo)

# Objectives

- ▶ Discussing some of the fundamental concepts of use case  
▶ مناقشة بعض المفاهيم الأساسية لحالة الاستخدام
- ▶ Modeling: use case, actor, association relationship, includes relationship, extends relationship, generalization relationship, flow of events, activity diagram, and Use Case diagram.  
▶ النمذجة: حالة الاستخدام ، الفاعل ، علاقة الارتباط ، تشمل العلاقة ، تمتد العلاقة ، علاقة التعميم وتدفق الأحداث ومخطط النشاط ومخطط حالة الاستخدام.
- ▶ Then, we'll look at how to model each of these in Rose.  
▶ بعد ذلك ، سننظر في كيفية تصميم كل منها في Rose.

# Use Case Modeling

- ▶ THE BEHAVIOR OF the system under development (i.e., what functionality must be provided by the system) is documented in a use case model
- ▶ تم توثيق سلوك النظام قيد التطوير (أي الوظائف التي يجب أن يوفرها النظام) في نموذج حالة الاستخدام
- ▶ Use case model illustrates the system's intended functions (use cases), its surroundings (actors), and relationships between the use cases and actors (use case diagrams).
- ▶ يوضح نموذج حالة الاستخدام الوظائف المقصودة للنظام (حالات الاستخدام) ، ومحيطه (الجهات الفاعلة) ، والعلاقات بين حالات الاستخدام والجهات الفاعلة (استخدم مخططات الحالة).
- ▶ The most important role of a use case model is for communication between the customers or end users and the developers to discuss the system's functionality and behavior.
- ▶ يتمثل الدور الأكثر أهمية لنموذج حالة الاستخدام في التواصل بين العملاء أو المستخدمين النهائيين والمطورين لمناقشة وظائف وسلوك النظام.
- ▶ The use case model starts in the Inception Phase with the identification of actors and principal use cases for the system.
- ▶ يبدأ نموذج حالة الاستخدام في مرحلة التأسيس بتحديد الفاعلين وحالات الاستخدام الرئيسية للنظام.
- ▶ The model is then matured in the Elaboration Phase, more detailed information is added to the identified use cases, and additional use cases are added if needed.
- ▶ ثم ينضج النموذج في مرحلة التفصيل ، وتضاف معلومات أكثر تفصيلاً إلى حالات الاستخدام المحددة ، وتضاف حالات استخدام إضافية إذا لزم الأمر.

# Use Case Modeling Concepts -Actors

## ► Definitionتعريف

- An actor is anyone or anything that interacts with the system being built.
- الفاعل هو أي شخص أو شيء يتفاعل مع النظام الذي يتم بناؤه.
- Actors are anything that is outside the system's scope.
- الفاعلون هم أي شيء خارج نطاق النظام.

## ► Descriptionوصف

- There are three primary types of actors:
  - هناك ثلاثة أنواع أساسية من الممثلين:
    - **Users of the system:** These are the most common actors, and are present in just about every system (actors = roles)
    - **مستخدمو النظام:** هؤلاء هم الممثلون الأكثر شيوعًا ، وهم موجودون في كل نظام تقريبًا (الممثلون = الأدوار)
    - **Other systems** that will interact with the system being built
    - **أنظمة أخرى** التي ستتفاعل مع النظام الذي يتم بناؤه
    - **Time:** Time becomes an actor when the passing of a certain amount of time triggers some event in the system.
    - **وقت:** يصبح الوقت فاعلاً عندما يؤدي مرور فترة زمنية معينة إلى حدوث حدث ما في النظام.

In UML, actors are represented with stick figures:

في UML، يتم تمثيل الممثلين بأشكال ثابتة:



► An actor may يجوز للممثل

- Only input information to the system

▪ فقط إدخال المعلومات إلى النظام

- Only receive information from the system

▪ تلقي المعلومات فقط من النظام

- Input and receive information to and from the system

▪ إدخال واستقبال المعلومات من وإلى النظام

► Typically, these actors are found in the problem statement and by conversations with customers and domain experts. The following questions may be used to help identify the actors for a system:

► عادةً ما يتم العثور على هؤلاء الممثلين في بيان المشكلة ومن خلال المحادثات مع العملاء وخبراء المجال. يمكن استخدام الأسئلة التالية للمساعدة في تحديد الجهات الفاعلة في النظام:

- Who is interested in a certain requirement?

▪ من يهتم بشرط معين؟

- Where in the organization is the system used?

▪ أين يتم استخدام النظام في المنظمة؟

- Who will benefit from the use of the system?

▪ من الذي سيستفيد من استخدام النظام؟

- Who will supply the system with this information, use this information, and remove this information?
  - من الذي سيزود النظام بهذه المعلومات ، ويستخدم هذه المعلومات ، ويزيل هذه المعلومات؟
- Who will support and maintain the system?
  - من سيدعم النظام ويحافظ عليه؟
- Does the system use an external resource?
  - هل يستخدم النظام مصدر خارجي؟
- Does one person play several different roles?
  - هل يلعب شخص واحد عدة أدوار مختلفة؟
- Do several people play the same role?
  - هل يلعب العديد من الأشخاص نفس الدور؟
- Does the system interact with a legacy system?
  - هل يتفاعل النظام مع نظام قديم؟



# What Constitutes a "Good" Actor?

- ▶ Care must be taken when identifying the actors for a system. This identification is done in an iterative fashion—the first cut at the list of actors for a system is rarely the final list.
- ▶ يجب توخي الحذر عند تحديد الجهات الفاعلة في النظام. يتم هذا التعريف بطريقة تكرارية - نادرًا ما يكون المقطع الأول في قائمة الممثلين لنظام ما هو القائمة النهائية.
- ▶ For example, is a new student a different actor than a returning student?
- ▶ على سبيل المثال ، هل الطالب الجديد هو ممثل مختلف عن الطالب العائد؟
- ▶ Suppose you initially say the answer to this question is yes. The next step is to identify how the actor interacts with the system.
- ▶ لنفترض أنك قلت في البداية أن الإجابة على هذا السؤال هي نعم. الخطوة التالية هي تحديد كيفية تفاعل الفاعل مع النظام.
- ▶ If the new student uses the system differently than the returning student, they are different actors. If they use the system in the same way, they are the same actor.
- ▶ إذا كان الطالب الجديد يستخدم النظام بشكل مختلف عن الطالب العائد ، فهم ممثلون مختلفون. إذا استخدموا النظام بنفس الطريقة ، فهم نفس الفاعل.

# What Constitutes a "Good" Actor?

- ▶ Another example is the creation of an actor for every role a person may play. This may also be overkill.  
▶ مثال آخر هو إنشاء ممثل لكل دور قد يلعبه الشخص. قد يكون هذا أيضا مبالغة.
- ▶ A good example is a teaching assistant in the ESU Course Registration System. The teaching assistant takes classes and teaches classes.  
▶ وخير مثال على ذلك هو مدرس مساعد في نظام التسجيل في دورات ESU. يأخذ المساعد التدريسي دروسًا ويعلمها.
- ▶ The capabilities needed to select courses to take and to teach are already captured by the identification of functionality needed by the Student and the Professor actors. Therefore, there is no need for a Teaching Assistant actor.  
▶ إن القدرات اللازمة لاختيار الدورات التي يجب أن تأخذها وتدرّسها يتم التقاطها بالفعل من خلال تحديد الوظائف التي يحتاجها الطالب والأستاذ الفاعلون. لذلك ، ليست هناك حاجة إلى ممثل مساعد تدريس.
- ▶ By looking at the identified actors and documenting how they use the system, you will iteratively arrive at a good set of actors for the system.  
▶ من خلال النظر إلى الجهات الفاعلة المحددة وتوثيق كيفية استخدامهم للنظام ، ستصل بشكل متكرر إلى مجموعة جيدة من الممثلين للنظام.

# Actors in the ESU Course

► The previous questions were answered as follows:

## Registration System

تمت الإجابة على الأسئلة السابقة على النحو التالي: ►

- Students want to register for courses  
يرغب الطلاب في التسجيل في الدورات
- Professors want to select courses to teach  
يريد الأساتذة اختيار الدورات للتدريس
- The Registrar must create the curriculum and generate a catalog for the semester  
يجب على المسجل إنشاء المناهج وإنشاء فهرس للفصل الدراسي
- The Registrar must maintain all the information about courses, professors, and students  
يجب على المسجل الاحتفاظ بجميع المعلومات حول الدورات والأساتذة والطلاب
- The Billing System must receive billing information from the system  
يجب أن يتلقى نظام الفواتير معلومات الفوترة من النظام
- Based on the answers to the questions posed, the following actors have been identified: Student, Professor, Registrar, and the Billing System.  
بناءً على الإجابات على الأسئلة المطروحة ، تم تحديد الجهات الفاعلة التالية: الطالب ، الأستاذ ، المسجل ، ونظام الفواتير.

# Actor Documentation

- ▶ A brief description for each actor should be added to the model. The description should identify the role the actor plays while interacting with the system.
- ▶ يجب إضافة وصف موجز لكل ممثل إلى النموذج. يجب أن يحدد الوصف الدور الذي يلعبه الممثل أثناء تفاعله مع النظام.
- ▶ The actor descriptions for the ESU Course Registration System are:
  - ▶ أوصاف الممثل لنظام التسجيل في دورات ESU هي:
    - **Student**— a person who is registered to take classes at the University
      - طالب-شخص مسجل لأخذ دروس في الجامعة
    - **Professor**— a person who is certified to teach classes at the University
      - أستاذ-شخص معتمد لتدريس الفصول في الجامعة
    - **Registrar**— the person who is responsible for the maintenance of the ESU Course Registration System
      - المسجل-الشخص المسؤول عن صيانة نظام التسجيل في دورات ESU
    - **Billing System**— the external system responsible for student billing
      - نظام الفواتير-النظام الخارجي المسؤول عن فواتير الطلاب

# Use Case Modeling Concepts - Use Cases-1

## ► Definition تعريف

- Use cases are an implementation-independent, high-level view of what the user expects from the system.
- حالات الاستخدام هي طريقة عرض عالية المستوى ومستقلة عن التنفيذ لما يتوقعه المستخدم من النظام.
- Describes a task that a user can perform using the system.
- يصف مهمة يمكن للمستخدم القيام بها باستخدام النظام.
- In other words, a use case illustrates how someone might use the system.
- بعبارة أخرى ، توضح حالة الاستخدام كيف يمكن لشخص ما استخدام النظام.

## ► Description وصف

- Describes requirements for the system      يصف متطلبات النظام
- Task described by a use case is composed of activities
- تتكون المهمة الموصوفة في حالة الاستخدام من الأنشطة
- Use case can have different variations called scenarios
- يمكن أن يكون لحالة الاستخدام أشكال مختلفة تسمى السيناريوهات
- Should not be used for functional decomposition !
- لا ينبغي أن تستخدم للتحلل الوظيفي!
- Use cases focus on what the system should do, not how the system will do it.
- تركز حالات الاستخدام على ما يجب أن يفعله النظام ، وليس كيفية قيام النظام بذلك.
- Your collection of use cases should let the customers easily see, at a very high level, your entire system.
- يجب أن تتيح مجموعة حالات الاستخدام للعملاء رؤية نظامك بالكامل بسهولة على مستوى عالٍ جدًا.

A use case is represented by this symbol:



Purchase Ticket

يتم تمثيل حالة الاستخدام بواسطة هذا الرمز:

# Use Case Modeling

## Concepts - Use Cases-2

- The advantage of looking at a system with use cases

► ميزة النظر إلى نظام مع حالات الاستخدام

- Is the ability to dissociate the implementation of the system from the reason the system is there in the first place.
  - هي القدرة على فصل تنفيذ النظام عن سبب وجود النظام في المقام الأول.
- It helps you focus on what is truly important—meeting the customer's needs and expectations without being instantly overwhelmed by implementation details.
  - يساعدك على التركيز على ما هو مهم حقًا - تلبية احتياجات العميل وتوقعاته دون أن تطغى على الفور في تفاصيل التنفيذ.
- By looking at the use cases, the customer can see what functionality will be provided, and can agree to the system scope before the project goes any further.
  - من خلال النظر في حالات الاستخدام ، يمكن للعميل معرفة الوظيفة التي سيتم توفيرها ، ويمكنه الموافقة على نطاق النظام قبل أن يمضي المشروع إلى أبعد من ذلك.

# Use Case Modeling

- ▶ How do I go about finding the use cases      كيف يمكنني البحث عن حالات الاستخدام؟

## Concepts – Use Cases-3

- A good way to begin is to examine any documentation the customers have provided. For example, a high-level scope or vision document can frequently help you identify the use cases.

- طريقة جيدة للبدء هي فحص أي وثائق قدمها العملاء. على سبيل المثال ، يمكن أن يساعدك نطاق عالي المستوى أو وثيقة رؤية بشكل متكرر في تحديد حالات الاستخدام.
- Ask the question what will the system do that provides value to the outside world?
- اطرح السؤال ماذا سيفعل النظام ليقدم قيمة للعالم الخارجي؟
- For each stakeholder, ask questions such as:
  - لكل صاحب مصلحة ، اطرح أسئلة مثل:
    - ▶ What will the stakeholder need to do with the system?
      - ▶ ما الذي يجب على أصحاب المصلحة فعله بالنظام؟
    - ▶ Will the stakeholder need to maintain any information (create, read, update, delete)?
      - ▶ هل سيحتاج صاحب المصلحة إلى الاحتفاظ بأي معلومات (إنشاء ، قراءة ، تحديث ، حذف)؟
    - ▶ Does the stakeholder need to inform the system about any external events?
      - ▶ هل يحتاج أصحاب المصلحة إلى إبلاغ النظام بأي أحداث خارجية؟
    - ▶ Does the system need to notify the stakeholder about certain changes or events?
      - ▶ هل يحتاج النظام إلى إخطار أصحاب المصلحة بشأن بعض التغييرات أو الأحداث؟

# Use Case Modeling

- When you have the final list of use cases, how do you know if you've found them all? Some questions to ask are:

## Concepts - Use Cases-4

► عندما تكون لديك القائمة النهائية لحالات الاستخدام ، كيف تعرف أنك عثرت عليها جميعاً؟ بعض الأسئلة التي يجب طرحها هي:

- Is each functional requirement in at least one use case? If a requirement is not in a use case, it will not be implemented.
- هل كل متطلب وظيفي في حالة استخدام واحدة على الأقل؟ إذا لم يكن أحد المتطلبات في حالة استخدام ، فلن يتم تنفيذه.
- Have you considered how each stakeholder will be using the system?
- هل فكرت في كيفية استخدام كل صاحب مصلحة للنظام؟
- What information will each stakeholder be providing for the system?
- ما هي المعلومات التي سيوفرها كل صاحب مصلحة للنظام؟
- What information will each stakeholder be receiving from the system?
- ما هي المعلومات التي سيحصل عليها كل صاحب مصلحة من النظام؟
- Have you considered maintenance issues? Someone will need to start the system and shut it down.
- هل فكرت في قضايا الصيانة؟ سيحتاج شخص ما إلى بدء تشغيل النظام وإغلاقه.
- Have you identified all of the external systems with which the system will need to interact?
- هل حددت جميع الأنظمة الخارجية التي سيحتاج النظام إلى التفاعل معها؟
- What information will each external system be providing to the system and receiving from the system?
- ما هي المعلومات التي سيوفرها كل نظام خارجي للنظام ويستقبلها من النظام؟



# What Constitutes a "Good" Use Case?

- The rule of thumb that I apply is the following:

القاعدة الأساسية التي أطبقها هي التالية: ►

- A use case typically represents a major piece of functionality that is complete from beginning to end.
- تمثل حالة الاستخدام عادةً جزءاً رئيسياً من الوظائف يكتمل من البداية إلى النهاية.
- A use case must deliver something of value to an actor.
- يجب أن تقدم حالة الاستخدام شيئاً ذا قيمة للممثل.
- For example, in the ESU Course Registration System, the student must select the courses for a semester, the student must be added to the course offerings, and the student must be billed. Is this three use cases, or just one? I would make it one because the functionality represents what happens from beginning to end.
- على سبيل المثال ، في نظام تسجيل دورات ESU ، يجب على الطالب تحديد الدورات لفصل دراسي ، ويجب إضافة الطالب إلى عروض الدورة ، ويجب أن تتم محاسبة الطالب. هل هذه ثلاث حالات استخدام أم واحدة فقط؟ سأجعله واحداً لأن الوظيفة تمثل ما يحدث من البداية إلى النهاية.
- What good would the system be if a student was not added to the courses selected (or at least notified if the addition does not occur)? Or if the student was not billed (the University would not stay in business if all courses were free!)?
- ما فائدة النظام إذا لم تتم إضافة طالب إلى الدورات المختارة (أو على الأقل إخطاره في حالة عدم حدوث الإضافة)؟ أو إذا لم يتم إصدار فاتورة للطالب (ستقوم الجامعة لا تبقى في العمل إذا كانت جميع الدورات مجانية!)?

# What Constitutes a "Good" Use Case?

- ▶ Another problem is how to bundle functionality that is different but seems to belong together.

▶ هناك مشكلة أخرى تتمثل في كيفية تجميع وظائف مختلفة ولكن يبدو أنها تنتمي معًا.

- ▶ For example, the Registrar must add courses, delete courses, and modify courses. Three use cases or one use case? Here again, I would make one use case—the maintenance of the curriculum, since the functionality is started by the same actor (the Registrar) and deals with the same entities in the system (the curriculum).

▶ على سبيل المثال ، يجب على المسجل إضافة الدورات وحذف الدورات وتعديل الدورات التدريبية. ثلاث حالات استخدام أم حالة استخدام واحدة؟ هنا مرة أخرى ، أود أن أقدم حالة استخدام واحدة - صيانة المنهج ، حيث أن الوظيفة تبدأ من قبل نفس الفاعل (المسجل) وتتعامل مع نفس الكيانات في النظام (المنهج).

# Use Cases in the ESU Course Registration System

► The following needs must be addressed by the system:

► يجب على النظام تلبية الاحتياجات التالية:

- The Student actor needs to use the system to register for courses.  
■ يحتاج ممثل الطالب إلى استخدام النظام للتسجيل في الدورات.
- After the course selection process is completed, the Billing System must be supplied with billing information.  
■ بعد اكتمال عملية اختيار الدورة التدريبية ، يجب تزويد نظام الفواتير بمعلومات الفواتير.
- The Professor actor needs to use the system to select the courses to teach for a semester, and must be able to receive a course roster from the system.  
■ يحتاج البروفيسور الفاعل إلى استخدام النظام لاختيار الدورات للتدريس لفصل دراسي ، ويجب أن يكون قادرًا على تلقي قائمة الدورة من النظام.
- The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the professors needed by the system.  
■ المسجل هو المسؤول عن إنشاء كتالوج الدورة لفصل دراسي ، والحفاظ على جميع المعلومات حول المناهج والطلاب والأساتذة التي يحتاجها النظام.

# Use Cases in the ESU Course Registration System

- Based on these needs, the following use cases have been identified:

■ بناءً على هذه الاحتياجات ، تم تحديد حالات الاستخدام التالية:

- |                                  |                          |
|----------------------------------|--------------------------|
| ▶ Register for courses           | التسجيل في الدورات       |
| ▶ Select courses to teach        | حدد الدورات للتدريس      |
| ▶ Request course roster          | طلب قائمة الدورة         |
| ▶ Maintain course information    | احتفظ بمعلومات الدورة    |
| ▶ Maintain professor information | حافظ على معلومات الأستاذ |
| ▶ Maintain student information   | حافظ على معلومات الطالب  |
| ▶ Create course catalog          | إنشاء كتالوج الدورة      |

# Brief Description of a Use Case

- ▶ The brief description of a use case states the purpose of the use case in a few sentences, providing a high-level definition of the functionality provided by the use case.

▶ يوضح الوصف المختصر لحالة الاستخدام الغرض من حالة الاستخدام في بضع جمل ، مما يوفر تعريفاً عالي المستوى للوظيفة التي توفرها حالة الاستخدام.

- ▶ The brief description of the Register for Courses use case is as follows:

▶ الوصف الموجز لحالة استخدام التسجيل في الدورات هو كما يلي:

- This use case is started by the Student. It provides the capability to create, modify, and/or review a student schedule for a specified semester.

■ حالة الاستخدام هذه بدأها الطالب. يوفر القدرة على إنشاء وتعديل و / أو مراجعة جدول الطالب لفصل دراسي محدد.

# The Flow of Events for a Use Case

- ▶ Each use case also is documented with a flow of events.
  - ▶ يتم أيضاً توثيق كل حالة استخدام مع تدفق الأحداث.
- ▶ The flow of events for a use case is a description of the events needed to accomplish the required behavior of the use case.
  - ▶ إن تدفق الأحداث لحالة الاستخدام هو وصف للأحداث اللازمة لإنجاز السلوك المطلوب لحالة الاستخدام.
- ▶ The flow of events is written in terms of what the system should do, not how the system does it. That is, it is written in the language of the domain, not in terms of implementation.
  - ▶ يتم كتابة تدفق الأحداث من حيث ما يجب أن يفعله النظام ، وليس كيف يفعله النظام. أي أنه مكتوب بلغة المجال وليس من حيث التطبيق.
- ▶ The flow of events should include:
  - ▶ يجب أن يشمل تدفق الأحداث ما يلي:
    - When and how the use case starts and ends
      - متى وكيف تبدأ حالة الاستخدام وتنتهي
    - What interaction the use case has with the actors
      - ما هو تفاعل حالة الاستخدام مع الممثلين
    - What data is needed by the use case
      - ما هي البيانات التي تحتاجها حالة الاستخدام
    - The normal sequence of events for the use case
      - التسلسل الطبيعي للأحداث لحالة الاستخدام
    - The description of any alternate or exceptional flows
      - وصف أي تدفقات بديلة أو استثنائية

# Use Case Specification Document

- ▶ The flow of events for a use case is contained in a document called the Use Case Specification.  
▶ يتم تضمين تدفق الأحداث لحالة الاستخدام في وثيقة تسمى مواصفات حالة الاستخدام.
- ▶ Each project should use a standard template for the creation of the Use Case Specification. I use the template from the Rational Unified Process.
- ▶ يجب أن يستخدم كل مشروع نموذجًا قياسيًا لإنشاء مواصفات حالة الاستخدام. أنا أستخدم القالب من عملية راشيونال الموحدة.

▶ 1.0 Use Case Name	1.0 اسم حالة الاستخدام	▶ 3.x < Special Requirement x > x	3.x < المتطلبات الخاصة
▶ 1.1 Brief Description	1.1 وصف موجز	▶ 4.0 Preconditions	4.0 شروط مسبقة
▶ 2.0 Flow of Events	2.0 تدفق الأحداث	▶ 4.x < Precondition x > x	4.x < الشرط المسبق
▶ 2.1 Basic Flow	2.1 التدفق الأساسي	▶ 5.0 Post Conditions	5.0 شروط النشر
▶ 2.2 Alternate Flows	2.2 التدفقات البديلة	▶ 5.x < Postcondition x > x	5.x < الشرط اللاحق
▶ 2.2.x < Alternate Flow x > x	2.2.x < التدفق البديل	▶ 6.0 Extension Points	6.0 نقاط الامتداد
▶ 3.0 Special Requirements	3.0 المتطلبات الخاصة	6.x < Extension Point x > x	6.x < نقطة التمديد

# Use Case Specification for the Select Courses to Teach Use Case

## ► 1.0 Use Case Name :Select Courses to Teach.

► ١,٠ اسم حالة الاستخدام: حدد الدورات التي تريد تدريسها.

## ► 1.1 Brief Description ١,١ وصف موجز

- This use case is started by the Professor. It provides the capability for the Professor to select up to four courses to teach for a selected semester.

■ حالة الاستخدام هذه بدأها الأستاذ. يوفر القدرة للأستاذ لاختيار ما يصل إلى أربع دورات للتدريس لفصل دراسي محدد.

## ► 2.0 Flow of Events ٢,٠ تدفق الأحداث

### ■ 2.1 Basic Flow ٢,١ التدفق الأساسي

- This use case begins when the Professor logs onto the Registration System and enters his/her password. The system verifies that the password is valid (if the password is invalid, Alternate Flow 2.2.1 is executed) and prompts the Professor to select the current semester or a future semester (if an invalid semester is entered, Alternate Flow 2.2.2 is executed).

■ تبدأ حالة الاستخدام هذه عندما يقوم الأستاذ بتسجيل الدخول إلى نظام التسجيل وإدخال كلمة المرور الخاصة به. يتحقق النظام من أن كلمة المرور صالحة (إذا كانت كلمة المرور غير صالحة ، يتم تنفيذ Alternate Flow 2.2.1) ويطلب من الأستاذ تحديد الفصل الدراسي الحالي أو الفصل الدراسي المقبل (إذا كان غير صالحتم إدخال الفصل الدراسي ، ويتم تنفيذ التدفق البديل ٢,٢,٢).



- The Professor enters the desired semester. The system prompts the Professor to select the desired activity: *ADD, DELETE, REVIEW, PRINT, or QUIT.*
- يدخل الأستاذ الفصل الدراسي المطلوب. يقوم النظام بمطالبة الأستاذ باختيار النشاط المطلوب: إضافة ، حذف ، مراجعة ، طباعة ، أو يترك.
- If the activity selected is ADD, the system displays the course screen containing a field for a course name and number. The Professor enters the name and number of a course (if an invalid name/number combination is entered, Alternate Flow 2.2.3 is executed). The system displays the course offerings for the entered course (if the course name cannot be displayed, Alternate Flow 2.2.4 is executed). The Professor selects a course offering. The system links the Professor to the selected course offering (if the link cannot be created, Alternate Flow 2.2.5 is executed). The use case then begins again.
- إذا كان النشاط المحدد هو ADD، فسيعرض النظام شاشة الدورة التي تحتوي على حقل لاسم الدورة التدريبية ورقمها. يقوم الأستاذ بإدخال اسم ورقم الدورة التدريبية (إذا تم إدخال مجموعة اسم / رقم غير صالحة ، فسيتم تنفيذ Alternate Flow 2.2.3). يعرض النظام عروض الدورة للدورة التي تم إدخالها (إذا تعذر عرض اسم الدورة التدريبية ، فسيتم تنفيذ Alternate Flow 2.2.4). يختار الأستاذ عرضاً للدورة. يربط النظام الأستاذ بعرض الدورة المحددة (إذا تعذر إنشاء الرابط ، فسيتم تنفيذ Alternate Flow 2.2.5). ثم تبدأ حالة الاستخدام مرة أخرى.

- If the activity selected is DELETE, the system displays the course offering screen containing a field for a course offering name and number. The Professor enters the name and number of a course offering (if an invalid name/number combination is entered, Alternate Flow 2.2.3 is executed). The system removes the link to the Professor (if the link cannot be removed, Alternate Flow 2.2.6 is executed). The use case then begins again.

► إذا كان النشاط المحدد هو DELETE، فسيعرض النظام شاشة عرض الدورة التي تحتوي على حقل لاسم ورقم عرض الدورة. يقوم الأستاذ بإدخال اسم ورقم عرض الدورة التدريسية (إذا تم إدخال مجموعة اسم / رقم غير صالحة ، فسيتم تنفيذ Alternate Flow 2.2.3). يقوم النظام بإزالة الارتباط إلى الأستاذ (إذا تعذر إزالة الارتباط ، فسيتم تنفيذ Alternate Flow 2.2.6). ثم تبدأ حالة الاستخدام مرة أخرى.

- If the activity selected is REVIEW, the system retrieves (if the course information cannot be retrieved, Alternate Flow 2.2.7 is executed) and displays the following information for all course offerings for which the Professor is assigned: course name, course number, course offering number, days of the week, time, and location. When the Professor indicates that he or she is through reviewing, the use case begins again.

► إذا كان النشاط المحدد هو REVIEW (مراجعة) ، يقوم النظام باسترداد (إذا تعذر استرداد معلومات الدورة التدريبية ، يتم تنفيذ Alternate Flow 2.2.7) ويعرض المعلومات التالية لجميع عروض الدورة التدريبية التي تم تعيين الأستاذ لها: اسم الدورة التدريبية ، ورقم الدورة التدريبية ، والدورة التدريبية عرض العدد وأيام الأسبوع والوقت والموقع. عندما يشير الأستاذ إلى أنه يخضع لعملية المراجعة ، تبدأ حالة الاستخدام مرة أخرى.

- If the activity selected is PRINT, the system prints the professor's schedule (if the schedule cannot be printed, Alternate Flow 2.2.8 is executed). The use case begins again.

► إذا كان النشاط المحدد هو PRINT ، فسيقوم النظام بطباعة جدول الأستاذ (إذا تعذر طباعة الجدول ، يتم تنفيذ التدفق البديل ٢,٢,٨). تبدأ حالة الاستخدام مرة أخرى.

- If the activity selected is QUIT, the use case ends.

► إذا كان النشاط المحدد هو QUIT ، تنتهي حالة الاستخدام.

## ► 2.2 Alternate Flows      ٢,٢ التدفقات البديلة

### ► 2.2.1 Invalid Password      ٢,٢,١ كلمة المرور غير صحيحة

- An invalid password is entered. The user can re-enter a password or terminate the use case.

► تم إدخال كلمة مرور غير صالحة. يمكن للمستخدم إعادة إدخال كلمة المرور أو إنهاء حالة الاستخدام.

- 2.2.2 Invalid Semester: The system informs the user that the semester is invalid. The user can re-enter the semester or terminate the use case.

► ٢,٢,٢ الفصل الدراسي غير صالح: يقوم النظام بإعلام المستخدم بأن الفصل الدراسي غير صالح. يمكن للمستخدم إعادة دخول الفصل الدراسي أو إنهاء حالة الاستخدام.

- 2.2.3 Invalid Course Name/Number: The system informs the user that the course name/number is invalid. The user can re- enter a valid name/number combination or terminate the use case.

► ٢,٢,٣ اسم / رقم الدورة غير صالح: يقوم النظام بإعلام المستخدم بأن اسم / رقم الدورة غير صالح. يمكن للمستخدم إعادة إدخال مجموعة اسم / رقم صالحة أو إنهاء حالة الاستخدام.

- ▶ 2.2.4 Course Offerings Cannot Be Displayed: The user is informed that this option is not available at the current time. The use case begins again.  
▶ ٢,٢,٤ لا يمكن عرض عروض الدورة التدريبية: يتم إبلاغ المستخدم أن هذا الخيار غير متوفر في الوقت الحالي. تبدأ حالة الاستخدام مرة أخرى.
- ▶ 2.2.5 Cannot Create Link Between Professor and Course Offering: The information is saved and the system will create the link at a later time. The use case begins again.  
▶ ٢,٢,٥ لا يمكن إنشاء ارتباط بين الأستاذ وطرح الدورة التدريبية: يتم حفظ المعلومات وسيقوم النظام بإنشاء الارتباط في وقت لاحق. تبدأ حالة الاستخدام مرة أخرى.
- ▶ 2.2.6 Link Between Professor and Course Offering Cannot Be Removed: The information is saved and the system will remove the link at a later time. The use case begins again.  
▶ ٢,٢,٦ لا يمكن إزالة الارتباط بين الأستاذ وطرح الدورة التدريبية: يتم حفظ المعلومات وسيقوم النظام بإزالة الارتباط في وقت لاحق. تبدأ حالة الاستخدام مرة أخرى.
- ▶ 2.2.7 Schedule Information Cannot Be Retrieved: The user is informed that this option is not available at the current time. The use case begins again.  
▶ ٢,٢,٧ لا يمكن استرجاع معلومات الجدول: يتم إبلاغ المستخدم أن هذا الخيار غير متوفر في الوقت الحالي. تبدأ حالة الاستخدام مرة أخرى.

- ▶ 2.2.8 Schedule Cannot Be Printed: The user is informed that this option is not available at the current time. The use case begins again.
  - ▶ ٢,٢,٨ يتعذر طباعة الجدول: يتم إبلاغ المستخدم أن هذا الخيار غير متوفر في الوقت الحالي. تبدأ حالة الاستخدام مرة أخرى.
- ▶ 3.0 Special Requirements المتطلبات الخاصة ٣,٠
  - There are no special requirements for this use case.
  - لا توجد متطلبات خاصة لحالة الاستخدام هذه.
- ▶ 4.0 Preconditions شروط مسبقة ٤,٠
  - 4.1 The Create Course Offerings subflow of the Maintain Course Information use case must execute before this use case begins.
  - ٤,١ يجب تنفيذ التدفق الفرعي لإنشاء عروض الدورة التدريبية لحالة استخدام "صيانة معلومات الدورة التدريبية" قبل أن تبدأ حالة الاستخدام هذه.
- ▶ 5.0 Post Conditions :There are no post conditions.
  - ▶ ٥,٠ شروط النشر: لا توجد شروط نشر.
- ▶ Extension Points نقاط الامتداد
  - There are no extension points. لا توجد نقاط تمديد.

# Paths and Scenarios

- ▶ Use Cases – abstractions      وقائع الاستخدام – التجريد
  - Really need **detailed interactions** – scenarios..      حقا بحاجة تفاعلات مفصلة-سيناريوهات
  - Scenarios can provide details of the interactions!      يمكن أن توفر السيناريوهات تفاصيل من التفاعلات!
  - Scenarios are instantiations of Use Cases:      السيناريوهات هي تمثيلات لحالات الاستخدام:  
[?] [?]      حالات الاستخدام مع البيانات
  - ▶ Use Cases with data
- ▶ Book provides two definitions for scenarios :      يقدم الكتاب تعريفين لـ سيناريوهات:
  - 1: merely an alternate path;      ١: مجرد مسار بديل ؛
  - 2: an instantiation of the use case with a specific path plus relevant data; the scenario is a realization of a use case (one possible path) with included data.
    - ٢: إنشاء مثيل لحالة الاستخدام بمسار محدد بالإضافة إلى البيانات ذات الصلة ؛ السيناريو هو ادر لحالة الاستخدام (مسار واحد محتمل) مع البيانات المضمنة.
- ▶ A **use case** describes how a class of users interacts with the system
- ▶ **حالة الاستخدام** يصف كيفية تفاعل فئة من المستخدمين مع النظام
- ▶ A **scenario** is a detailed, step-by-step sequence of interactions between a user and the system for one type of interaction
- ▶ **السيناريو** عبارة عن تسلسل تفصيلي خطوة بخطوة من التفاعلات بين المستخدم والنظام لنوع واحد من التفاعل
- ▶ **Scenarios can be used**      يمكن استخدام السيناريوهات
  - during requirements capture for feedback to users and analysts that the use cases accurately reflect user needs, and
  - أثناء التقاط المتطلبات للموقف على آرائهم للمستخدمين والمحللين أنتعكس حالات الاستخدام بدقة احتياجات المستخدم ، و
  - in testing to test whether the system reflects the requirements.
  - في الاختبار للاختبار ما إذا كان النظام يعكس المتطلبات.

# More – very important tidbits here.

- The relationship between requirements and use cases is subject of much discussion.

العلاقة بين المتطلبات وحالات الاستخدام موضع نقاش كبير. ►

- A use case describes a unit of behavior
  - تصف حالة الاستخدام وحدة من سلوك
- A requirement describes a law that governs behavior
  - مطلب يصف القانون الذي يحكم سلوك
- A use case can capture/satisfy/describe one or more functional requirements
  - يمكن لحالة الاستخدام أن تلتقط / ترضي / تصف واحدًا أو أكثر المتطلبات الوظيفية
- A functional requirement may be satisfied by one or more use cases...
  - يمكن استيفاء أحد المتطلبات الوظيفية من خلال حالة استخدام واحدة أو أكثر ...



# Use Case Modeling Concepts

## - Relationships

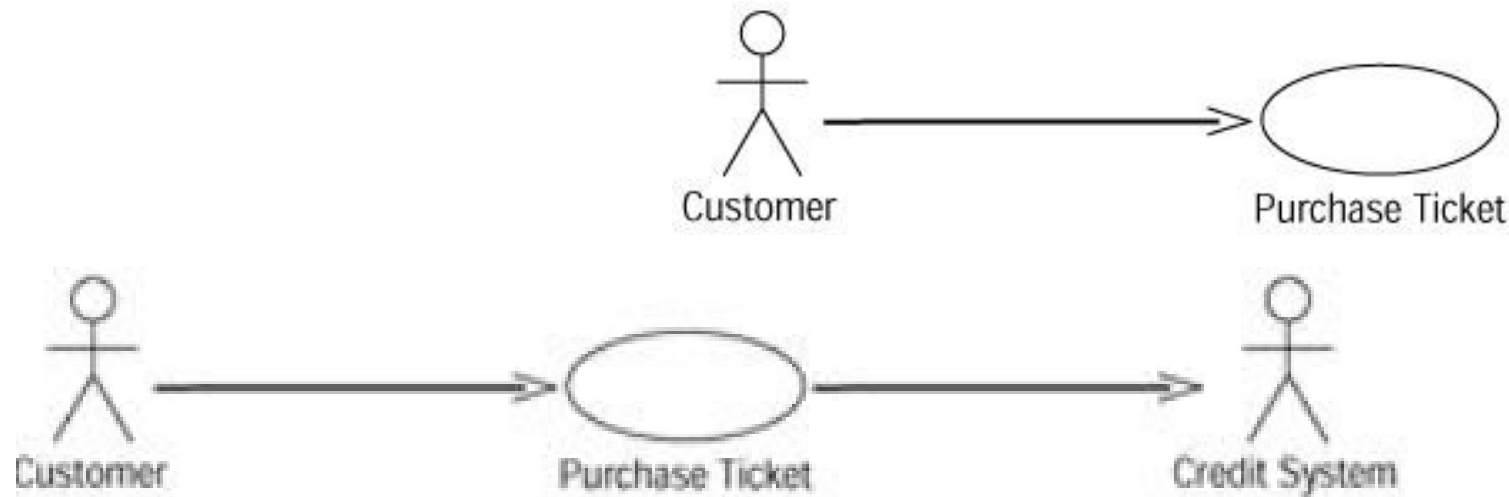
- ▶ The Association Relationship is used to show the relationship between a use case and an actor.  
▶ ال علاقة الجمعية تُستخدم لإظهار العلاقة بين واقعة الاستخدام والممثل.
- ▶ There are three types of relationships between use cases: an **includes relationship**, an **extends relationship**, and a **generalization relationship**.  
▶ هناك ثلاثة أنواع من العلاقات بين حالات الاستخدام: أيتضمن العلاقة، ويمتد العلاقة، وأعلاقة التعميم.
- ▶ There is only one relationship allowed between actors. This is a **generalization relationship**  
▶ هناك علاقة واحدة فقط مسموح بها بين الممثلين. هذا ال علاقة التعميم

# Use Case Modeling Concepts - Relationships

## ► Association Relationship علاقة الجمعية

- The relationship between an actor and a use case is an association relationship.

■ العلاقة بين الفاعل وحالة الاستخدام هي علاقة ارتباط.



- Although information flows in both directions—from the reservation system to the credit system and back again—the arrow indicates only who initiated the communication.
- على الرغم من تدفق المعلومات في كلا الاتجاهين - من نظام الحجز إلى نظام الائتمان والعودة مرة أخرى - يشير السهم فقط إلى من بدأ الاتصال.
- With the exception of use cases in includes and extends relationships, every use case must be initiated by an actor.

■ 1-32 باستثناء حالات الاستخدام في تضمين وتوسيع العلاقات، يجب أن يبدأ ممثل كل حالة استخدام.

# Use Case Modeling Concepts

## - Relationships

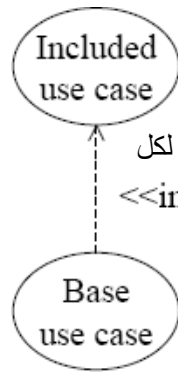
► Includes Relationship يشمل العلاقة

- An includes relationship allows one use case to use the functionality provided by another use case.

▪ تسمح علاقة التضمين لحالة استخدام واحدة باستخدام الوظيفة التي توفرها حالة استخدام أخرى.

- This relationship can be used in one of two cases. يمكن استخدام هذه العلاقة في إحدى حالتين.

- **First**, if two or more use cases have a large piece of functionality that is identical, this functionality can be split into its own use case. Each of the other use cases can then have an includes relationship with this new use case.



► أولاً، إذا كانت حالتنا استخدام أو أكثر تحتويان على وظيفة كبيرة متطابقة ، فيمكن تقسيم هذه الوظيفة إلى حالة الاستخدام الخاصة بها. يمكن أن يكون لكل

<<include>>

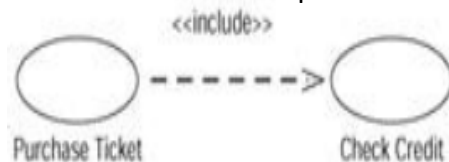
حالة من حالات الاستخدام الأخرى علاقة تضمين مع حالة الاستخدام الجديدة هذه.

- **The second** case where an includes relationship is helpful is a situation in which a single use case has an unusually large amount of functionality. An includes relationship can be used to model two smaller use cases instead.

► **الثاني** الحالة التي تكون فيها علاقة التضمين مفيدة هي الحالة التي تحتوي فيها حالة الاستخدام الفردي على قدر كبير من الوظائف بشكل غير عادي.

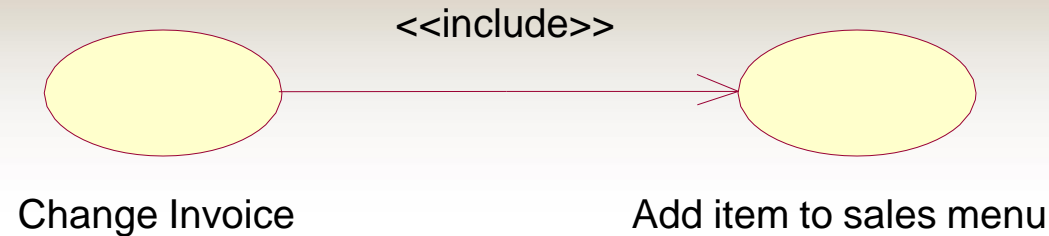
يمكن استخدام علاقة التضمين لنمذجة حالتي استخدام أصغر بدلاً من ذلك.

- Includes relationships are shown in Rose with dashed arrows and the word <<include>>, as in the following Figure



▪ يتم عرض العلاقات المتضمنة في Rose مع أسهم مقطوعة وكلمة <<include>>، كما في الشكل التالي

# Use Cases include other Use Cases



In this case the second Use Case is always invoked as part of the execution of the first  
في هذه الحالة ، يتم دائماً استدعاء واقعة الاستخدام الثانية كجزء من تنفيذ الحالة الأولى

The «includes» association always occurs when the use case which includes it occurs.  
يحدث الاقتران "include" دائماً عند حدوث حالة الاستخدام التي تتضمنها.

Include stereotype allows use case designers to avoid duplicating steps across multiple use cases (a reuse strategy).

يسمح تضمين الصورة النمطية لمصممي حالات الاستخدام بتجنب تكرار الخطوات عبر حالات الاستخدام المتعددة (استراتيجية إعادة الاستخدام).

Arrow extends away from the owning parent use case.

يمتد السهم بعيداً عن حالة استخدام الوالدين المالكين.

# Use Case Modeling Concepts - Relationships

## ► Extends Relationship يوسع العلاقة

- In extends relationship a use case is **optionally** extended by functionality of another use case

■ في تمديد العلاقة حالة الاستخدام اختياريًا من خلال وظيفة حالة استخدام أخرى

- An extend relationship is used to show

■ يتم استخدام علاقة التمديد للعرض

► Optional behavior سلوك اختياري

► Behavior that is run only under certain conditions such as triggering an alarm

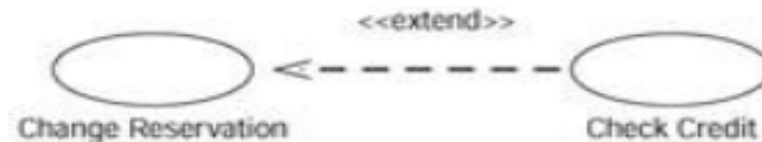
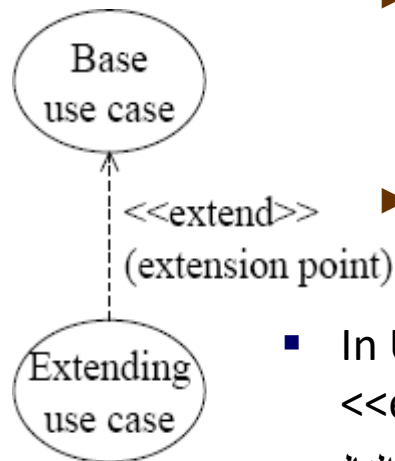
► السلوك الذي يتم تشغيله فقط في ظل ظروف معينة مثل إطلاق الإنذار

► Several different flows that may be run based on actor selection

► عدة تدفقات مختلفة يمكن تشغيلها بناءً على اختيار الممثل

- In UML, the extends relationship is shown as a dashed arrow with the word <<extend>>, as in the following Figure.

■ في UML، تظهر علاقة الامتداد كسهم متقطع بالكلمة <<extend>>، كما في الشكل التالي.



# Associations Between Use Cases

The occurrence of an «extends» association **depends on a true condition** in the use case which it extends.

حدوث اقتران «يمتد» يعتمد على حالة حقيقية في حالة الاستخدام التي تمتد.

A Use Case is only **called occasionally** from another Use Case

واقعة الاستخدام هي فقط تسمى من حين لآخر من واقعة استخدام أخرى

Extending use case **would not exist without** the use case it is extending (the extended use case);

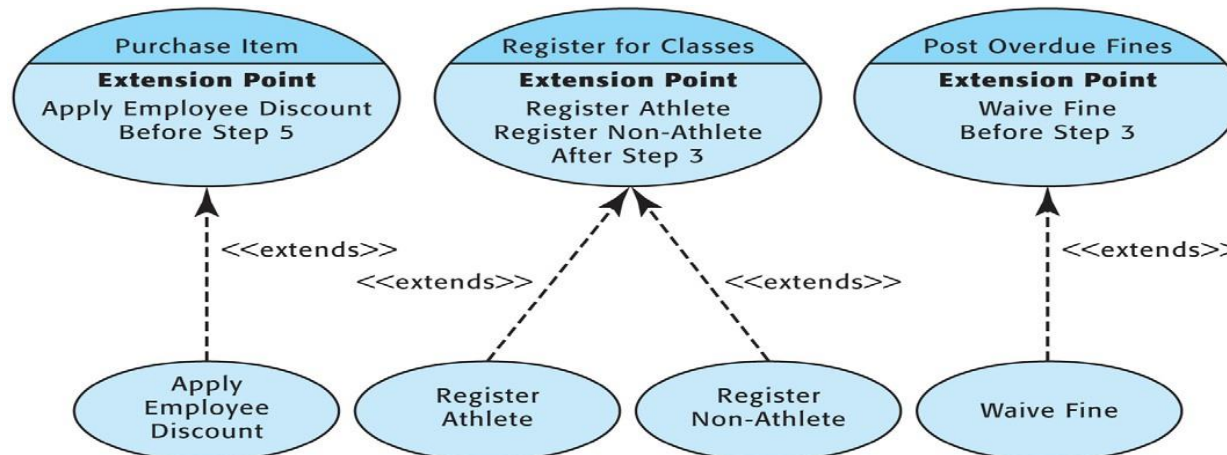
تمديد حالة الاستخدام لن توجد بدون حالة الاستخدام التي يتم تمديدتها (حالة الاستخدام الممتدة)؛

Used for **special cases; exceptional behaviors. Inserted behaviors**

مستخدم للحالات الخاصة سلوكيات استثنائية. السلوكيات المدرجة

Arrow **points toward** the base use case that is being extended...

سهم يشير إلى حالة الاستخدام الأساسية يتم تمديده ...

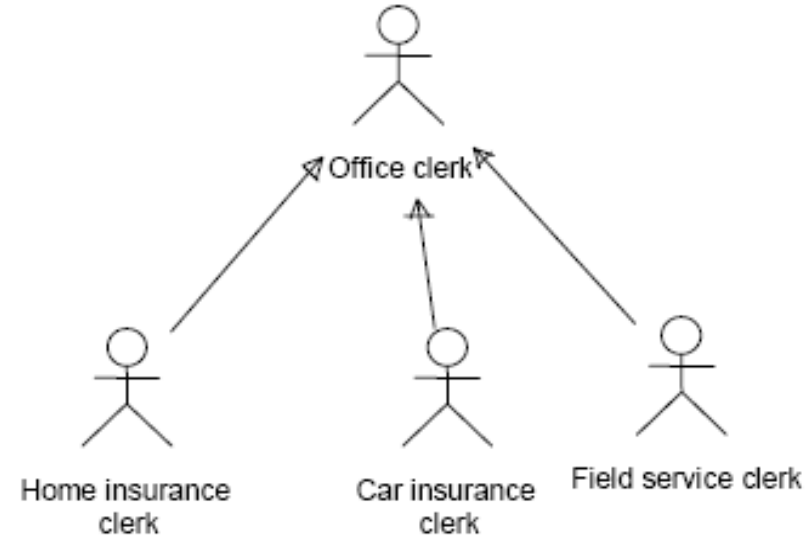
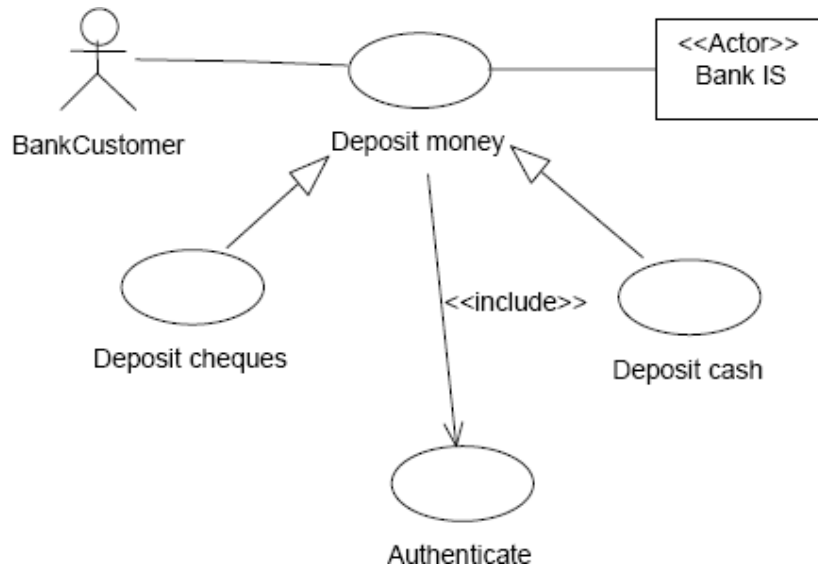
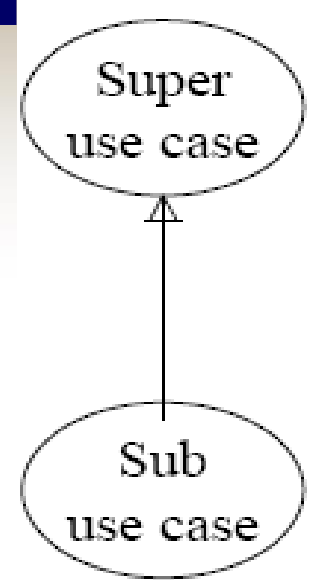


# Use Case Modeling Concepts

## ► Generalization Relationship علاقة التعميم

### - Relationships

- A generalization relationship is used to show that several actors or use cases have some commonality.
- تُستخدم علاقة التعميم لإظهار أن العديد من الجهات الفاعلة أو حالات الاستخدام لها بعض القواسم المشتركة.
- Sub use case inherits behavior and semantics from super use cases
- ترث حالة الاستخدام الفرعية السلوك والدلالات من حالات الاستخدام الفائقة
- Example of a generalization relationship
- مثال على علاقة التعميم



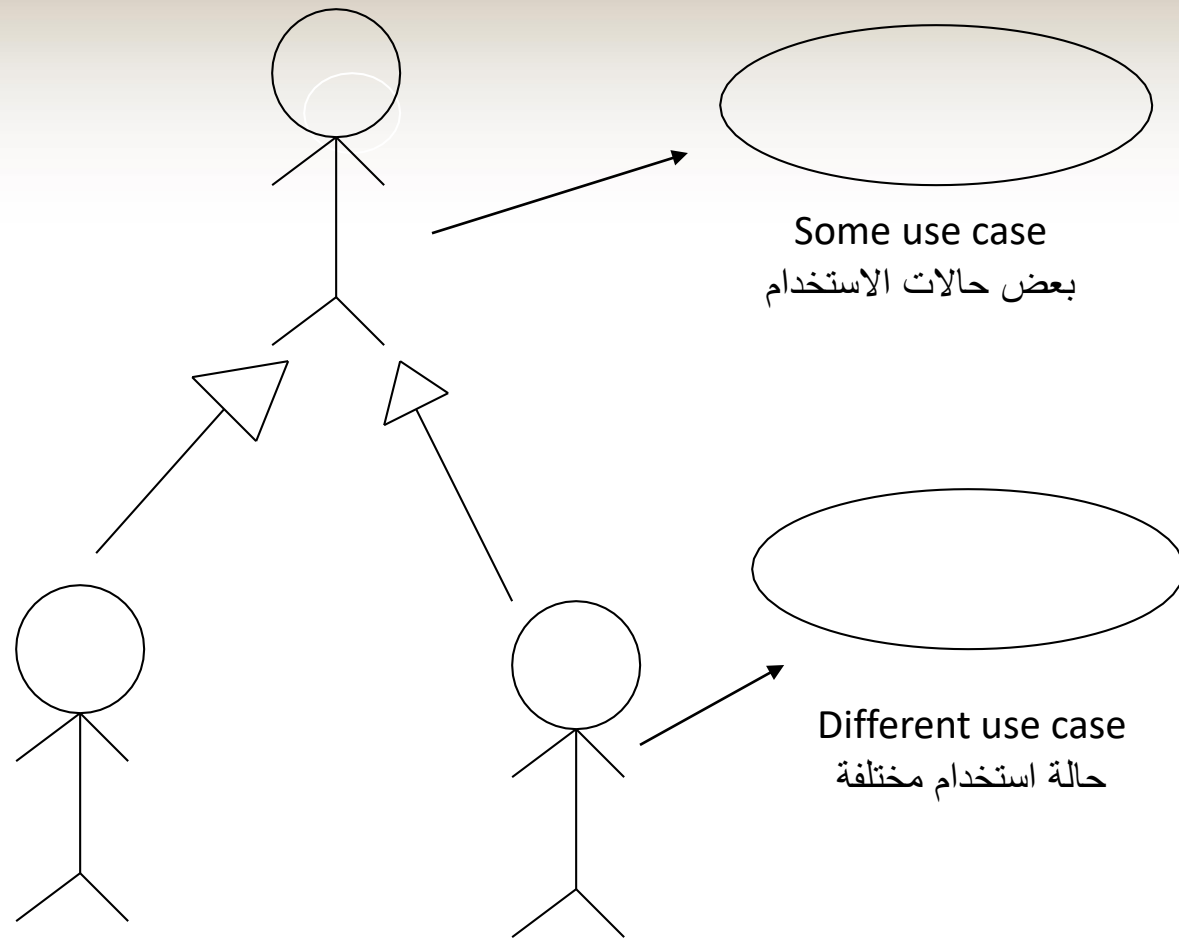
# Associations: Generalization in Actors

- Generalization: Abstract the commonalities.

► التعميم: الملخص القواسم المشتركة.

- two sub-actors generalized into a super-actor
  - اثنين من الممثلين الفرعيين المعمم في سوبر الفاعل
- Have both behavior and attributes in common – described under the super-actor.
  - لديك قواسم مشتركة في السلوك والسمات - يتم وصفها ضمن الممثل الخارق.
- Super-actor should interact with use cases when **ALL** of its sub-actors interact the same way;
  - ممثل خارق يجب أن تتفاعل مع حالات الاستخدام عندما **الجميع** الفاعلين الفرعيين تتفاعل بنفس الطريقة
- Sub-actors should interact with use cases when their individual interactions differ from that of the super-actor.
  - الجهات الفاعلة الفرعية يجب أن تتفاعل مع حالات الاستخدام عند تفاعلاتهم الفردية يختلف من الممثل الخارق.





# Use Case Diagrams

## ► Use Case Diagram

استخدم الرسم البياني

### ■ Definition

تعريف

- Shows the relationships between a set of use cases , the actors involved in these use cases and the relationships between them.

► يوضح العلاقات بين مجموعة من حالات الاستخدام والجهات الفاعلة المشاركة في حالات الاستخدام هذه والعلاقات فيما بينها.

### ■ Description

وصف

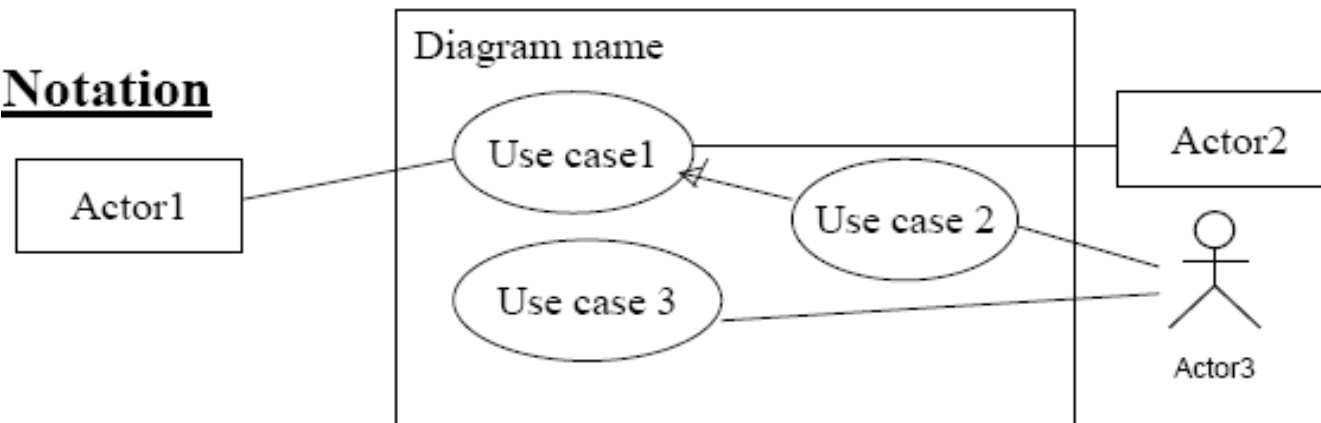
- Tool for requirement determination

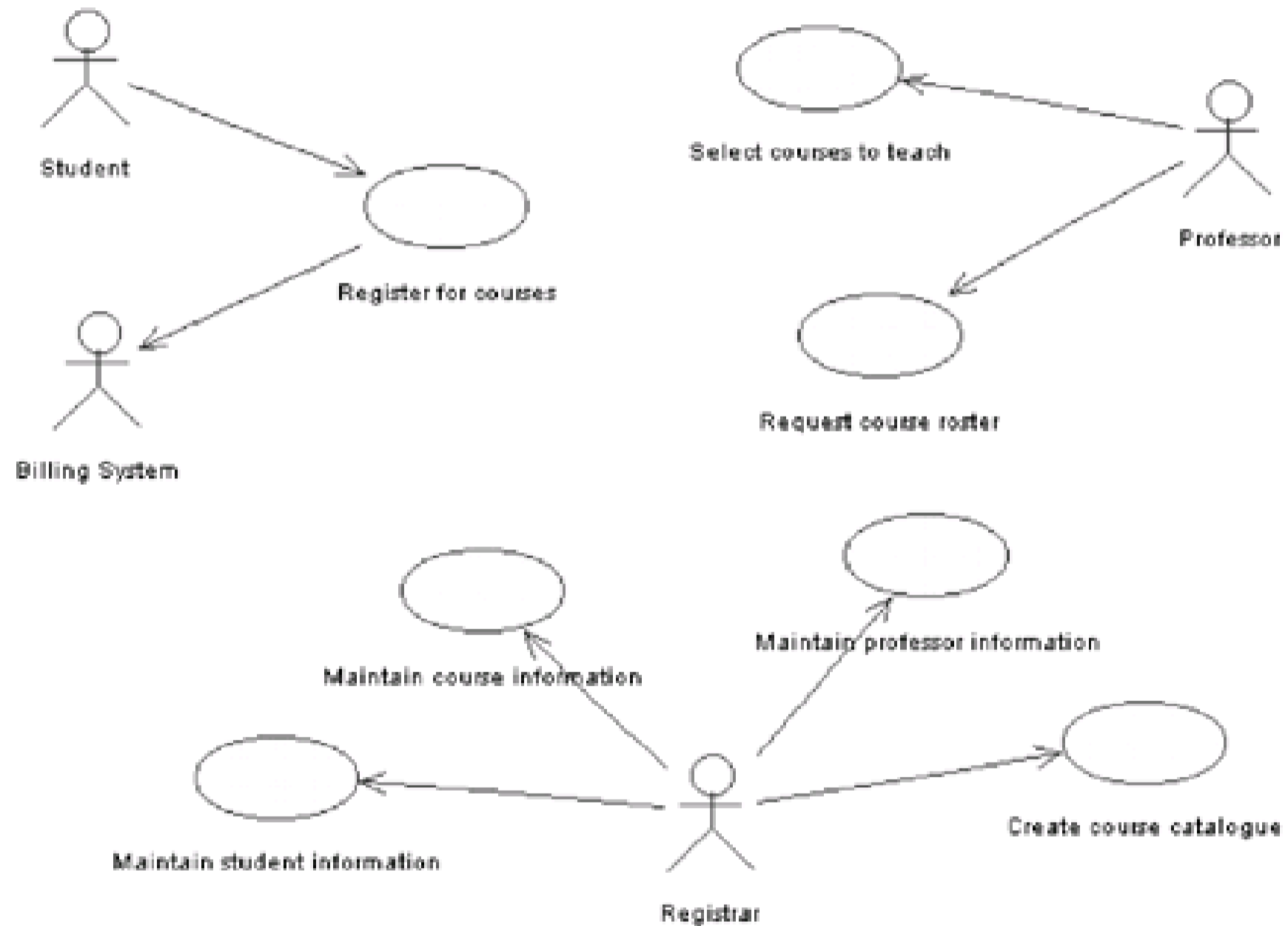
► أداة لتحديد المتطلبات

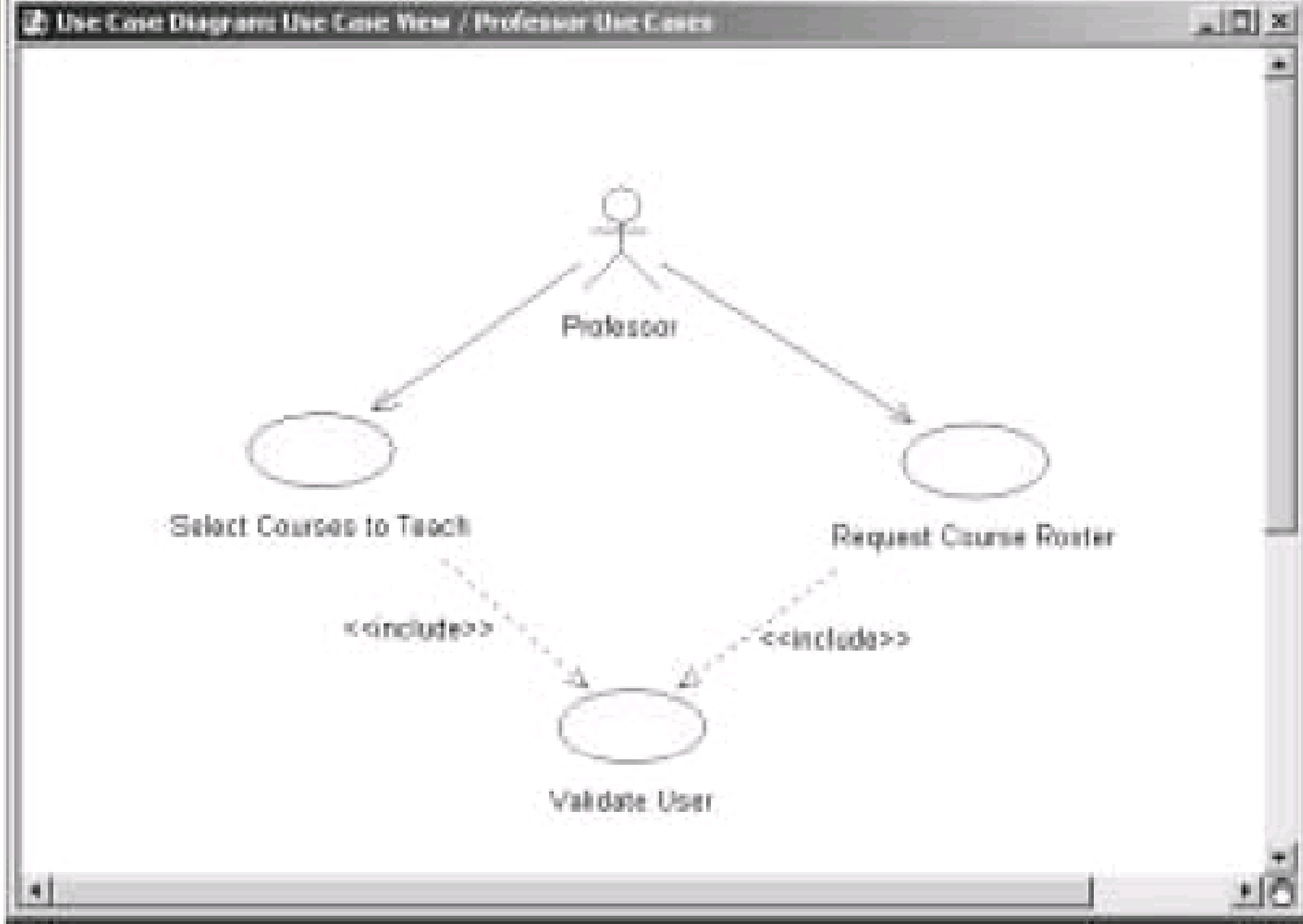
- Use case describes those activities which are to be supported by the software under development

► تصف حالة الاستخدام تلك الأنشطة التي سيتم دعمها بواسطة البرنامج قيد التطوير

## Notation








# Use Case Diagrams Cont'

- ▶ **Some things to keep in mind as you are creating Use Case diagrams include:**
  - ▶ تتضمن بعض الأشياء التي يجب وضعها في الاعتبار أثناء إنشاء مخططات حالة الاستخدام ما يلي:
    - Do not model actor-to-actor associations (although generalizations are OK).
      - لا تقم بنمذجة ارتباطات الممثل إلى الفاعل (على الرغم من أن التعميمات لا بأس بها).
    - Do not draw an association directly between two use cases (although includes or extends relationships are OK).
      - لا ترسم ارتباطًا مباشرًا بين حالتي استخدام (على الرغم من أن تضمين العلاقات أو توسيعها أمر جيد).
    - Every use case must be initiated by an actor, the exception here is an includes or extends relationship.
      - يجب أن تبدأ كل حالة استخدام من قبل أحد الممثلين ، والاستثناء هنا هو علاقة التضمين أو الامتداد.
    - Think of the database as a layer underneath the entire Use Case diagram. You can enter information in the database using one use case, and then access that information from the database in another use case. You don't have to draw associations from one use case to another to show information flow.
      - فكر في قاعدة البيانات كطبقة أسفل الرسم التخطيطي لحالة الاستخدام بالكامل. يمكنك إدخال المعلومات في قاعدة البيانات باستخدام حالة استخدام واحدة ، ثم الوصول إلى تلك المعلومات من قاعدة البيانات في حالة استخدام أخرى. لا يتعين عليك استخلاص ارتباطات من حالة استخدام إلى أخرى لإظهار تدفق المعلومات.

# Use case specification example

Use Case Number:	UC-03	
Use Case Name:	Edit Member Profile	
Actor (s):	Buyer, Seller 	
Maturity:	Focused	
Summary:	This use case is started by a <i>Buyer</i> or a <i>Seller</i> . It provides the capability for one of these actors to edit their <i>member profile</i> .	
Basic Course of Events:	<p>Actor Action</p> <p>1. This use case is started when a <i>Buyer</i> or <i>Seller</i> elects to edit their <i>member profile</i>. Perform S1-Login (<b>subflow – later</b>)</p> <p>3. The Actor updates their <i>member profile</i>.</p> <p>6. The Actor confirms that the information is correct. <b>{Profile Change}</b></p> <p>8. This use case concludes when the Actor receives visual confirmation of the update.</p>	<p>System Response</p> <p>2. The System displays the Actor's <i>member profile</i> and prompts the Actor to update it.</p> <p>ξ. The System validates the information entered by the Actor. <b>{Validate Information}</b> ο. The System prompts the Actor for confirmation.</p> <p>7. The System updates the Actor's <i>member profile</i> to the <i>member repository</i> and informs the Actor that the information was updated successfully.</p>

## Continuing....

<b>Alternate Paths:</b>	<b>A1 Change Member Profile</b> At <b>{Profile Change}</b> the <i>Member</i> indicates that he/she entered incorrect information. The System immediately returns to the step 2.
<b>Exception Paths:</b>	<b>E1 Handle Invalid Information</b> At <b>{Validate Information}</b> if any fields are entered incorrectly. the System indicates the fields that were entered incorrectly and prompts the <i>Buyer</i> or <i>Seller</i> to make the necessary corrections. If errors, the flow of events is resumed at Basic Flow Step 2.
<b>Extension Points:</b>	<b>{Change Profile }, {Validate Information}</b>
<b>Triggers:</b>	The <i>Buyer</i> or <i>Seller</i> would like to edit their <i>member profile</i> .
<b>Assumptions:</b>	The <i>Buyer</i> or <i>Seller</i> is aware of the steps required to edit their <i>member profile</i> .
<b>Preconditions:</b>	The System is functioning properly. Actor already has a Profile stored in the Profile Database???
<b>Post Conditions:</b>	The <i>member profile</i> was successfully updated to the <i>member repository</i> . Actor sent email to confirm changes?
<b>Reference - Business Rules:</b>	See Business Rules section: 2.3.1 and 2.3.5.
<b>Reference - Risks:</b>	See Risks List sections: 2.1 and 2.4.
<b>Author (s):</b>	<i>Team3</i>
<b>Date:</b>	11-04-04

# UML Activity Diagrams



# Activity Diagrams-1

- ▶ An activity diagram is another way to model the flow of events.  
▶ مخطط النشاط هو طريقة أخرى لنمذجة تدفق الأحداث.
- ▶ An activity diagram shows you the same information as a textual flow of events would.  
▶ يوضح لك مخطط النشاط نفس المعلومات مثل التدفق النصي للأحداث.
- ▶ We use activity diagrams in business modeling to depict the workflow through a business process.  
▶ نستخدم مخططات النشاط في نمذجة الأعمال لتصوير سير العمل من خلال عملية الأعمال.
- ▶ Here, we will use them to depict the flow through a piece of the system.  
▶ هنا ، سوف نستخدمها لتصوير التدفق عبر جزء من النظام.
- ▶ At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.  
▶ في هذه المرحلة من دورة الحياة ، يمكن إنشاء مخططات النشاط لتمثيل التدفق عبر حالات الاستخدام أو قد يتم إنشاؤها لتمثيل التدفق داخل حالة استخدام معينة.
- ▶ These diagrams represent the dynamics of the system.  
▶ تمثل هذه المخططات ديناميات النظام.

# Activity

**An activity diagram** shows activities and the flow of control and data between them.

مخطط النشاط يظهر الأنشطة وتدفق التحكم والبيانات فيما بينها.

**An activity** is a non-atomic task or procedure decomposable into actions.

نشاط هي مهمة أو إجراء غير ذري قابل للتحلل إلى أفعال.

**An action** is a task or procedure that cannot be broken into parts.

Actions are smaller steps that take place within an activity.

إجراء هي مهمة أو إجراء لا يمكن تقسيمه إلى أجزاء. الإجراءات هي خطوات أصغر تحدث داخل النشاط.

An activity is modeled using the following symbol:

يتم نمذجة النشاط باستخدام الرمز التالي:



# Activity Diagrams -3

## ► Actions they may occur at one of four times:

► الإجراءات التي قد تحدث في واحدة من أربع مرات:

- Upon entering the activity. An entry action occurs as soon as the activity begins, and is marked with the word "entry."  
■ عند دخول النشاط. يحدث إجراء الإدخال بمجرد بدء النشاط ، ويتم تمييزه بالكلمة "إدخال".
- When exiting the activity. An exit action occurs as you are leaving the activity, and is marked with the word "exit."  
■ عند الخروج من النشاط. يحدث إجراء الخروج أثناء مغادرتك للنشاط ، ويتم تمييزه بكلمة "خروج".
- While performing the activity. These actions occur while in the activity and continue until you leave the activity. They are marked with the word "do."  
■ أثناء أداء النشاط. تحدث هذه الإجراءات أثناء وجودك في النشاط وتستمر حتى تغادر النشاط. يتم تمييزها بكلمة "تفعل".
- Upon a specific event. These actions happen if and only if a specific event occurs. They are marked by the word "event," followed by the event name.  
■ في حدث معين. تحدث هذه الإجراءات في حالة حدوث حدث معين فقط في حالة حدوثه. يتم تمييزها بكلمة "حدث" ، متبوعة باسم الحدث.

### Display Available Flights

entry/ Find all flights for selected cities/dates

entry/ Determine flights with available seats

do/ Display list of flights with available seats

do/ Highlight flight with lowest fare

event/ User requests fare information/ Display fare information

# Activity Diagrams-4

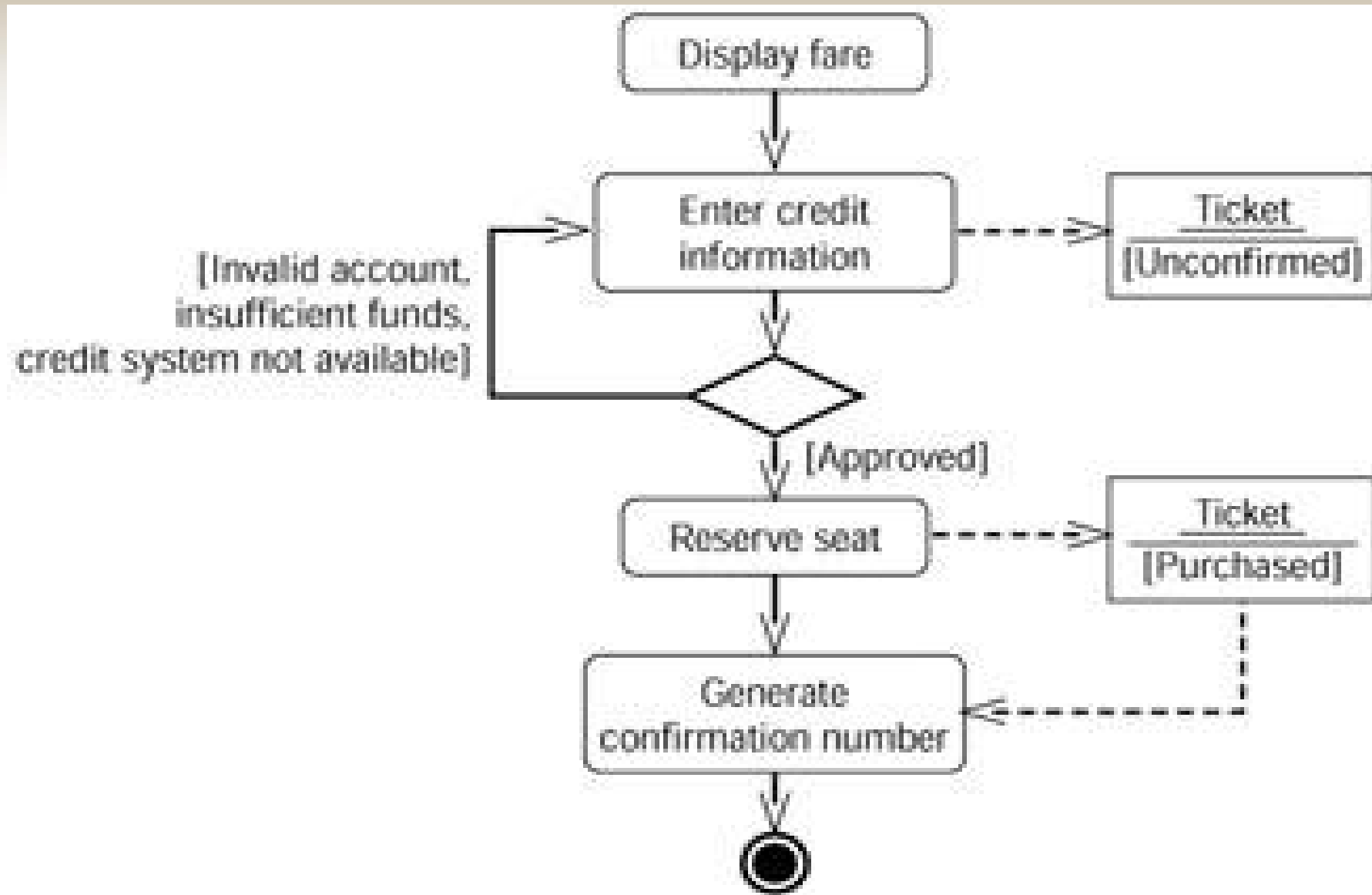
## ► Start and End States: حالات البداية والنهاية:

- Each activity diagram must have a start state, which is drawn as a solid dot, to signify where the flow begins.
- يجب أن يكون لكل مخطط نشاط حالة بداية ، يتم رسمها كنقطة صلبة ، للإشارة إلى مكان بدء التدفق.
- End states are optional on the diagram. They show you where the flow ends, and are represented by a bull's-eye.
- حالات النهاية اختيارية في الرسم التخطيطي. إنها توضح لك أين ينتهي التدفق ، وتمثلها عين الثور.
- You can have more than one end state on the diagram, but only a single start state.
- يمكن أن يكون لديك أكثر من حالة نهاية واحدة في الرسم التخطيطي ، ولكن فقط حالة بداية واحدة.

## ► Objects and Object Flows الكائنات وتدفق الكائنات

- An object is an entity that is affected by the flow. It may be used or changed by an activity in the flow.
- الكائن هو كيان يتأثر بالتدفق. يمكن استخدامه أو تغييره بواسطة نشاط في التدفق.
- On an activity diagram, you can display the object and its state so that you can understand where and how the object's state changes.
- في الرسم التخطيطي للنشاط ، يمكنك عرض الكائن وحالته بحيث يمكنك فهم أين وكيف تتغير حالة الكائن.
- Objects are linked to activities through object flows. An object flow is a dashed arrow drawn from an activity to the object it changes, or from the object to the activity that needs to use it.
- الكائنات مرتبطة بالأنشطة من خلال تدفقات الكائنات. تدفق الكائن هو سهم متقطع مرسوم من نشاط إلى الكائن الذي يغيره ، أو من الكائن إلى النشاط الذي يحتاج إلى استخدامه.

# Activity Diagrams-5



# Activity Diagrams-6

## ► Transitions      الانتقالات

- A transition shows how the flow of control moves from one activity to another
- يُظهر الانتقال كيف ينتقل تدفق التحكم من نشاط إلى آخر
- We can, however, set limitations on the transition to control when the transition occurs.
- ومع ذلك ، يمكننا وضع قيود على الانتقال للتحكم في وقت حدوث الانتقال.
- This can be done either by using an event or a guard condition. If an event is specified for a transition, the event must happen in order for the transition to occur. The transition arrow is labeled with the event name, along with any arguments in parenthesis.
- يمكن القيام بذلك إما باستخدام حدث أو شرط حارس. إذا تم تحديد حدث للانتقال ، فيجب أن يقع الحدث حتى يحدث الانتقال. يتم تسمية سهم الانتقال باسم الحدث ، جنبًا إلى جنب مع أي وسيطات بين قوسين.

# Activity Diagrams-7

## An event triggers a transition

حدث ما يطلق عملية انتقال

Here we can see that if the user changes their mind and performs a cancel event, the purchase price will be refunded and the ticket will be canceled.

هنا يمكننا أن نرى أنه إذا غير المستخدم رأيه وأجرى حدث إلغاء ، فسيتم رد سعر الشراء وسيتم إلغاء التذكرة.



## A guard condition controls whether or not the transition can occur

تتحكم حالة الحارس في إمكانية حدوث الانتقال أم لا

If a guard condition is present, it must be true in order for the transition to occur.

في حالة وجود شرط حارس ، يجب أن يكون صحيحًا حتى يحدث الانتقال.

In this example, a new confirmation number is needed only if there is a new reservation made. If we are changing an existing reservation, the old confirmation number will remain.

في هذا المثال ، يلزم إدخال رقم تأكيد جديد فقط في حالة إجراء حجز جديد. إذا قمنا بتغيير حجز حالي ، فسيظل رقم التأكيد القديم ساريًا.



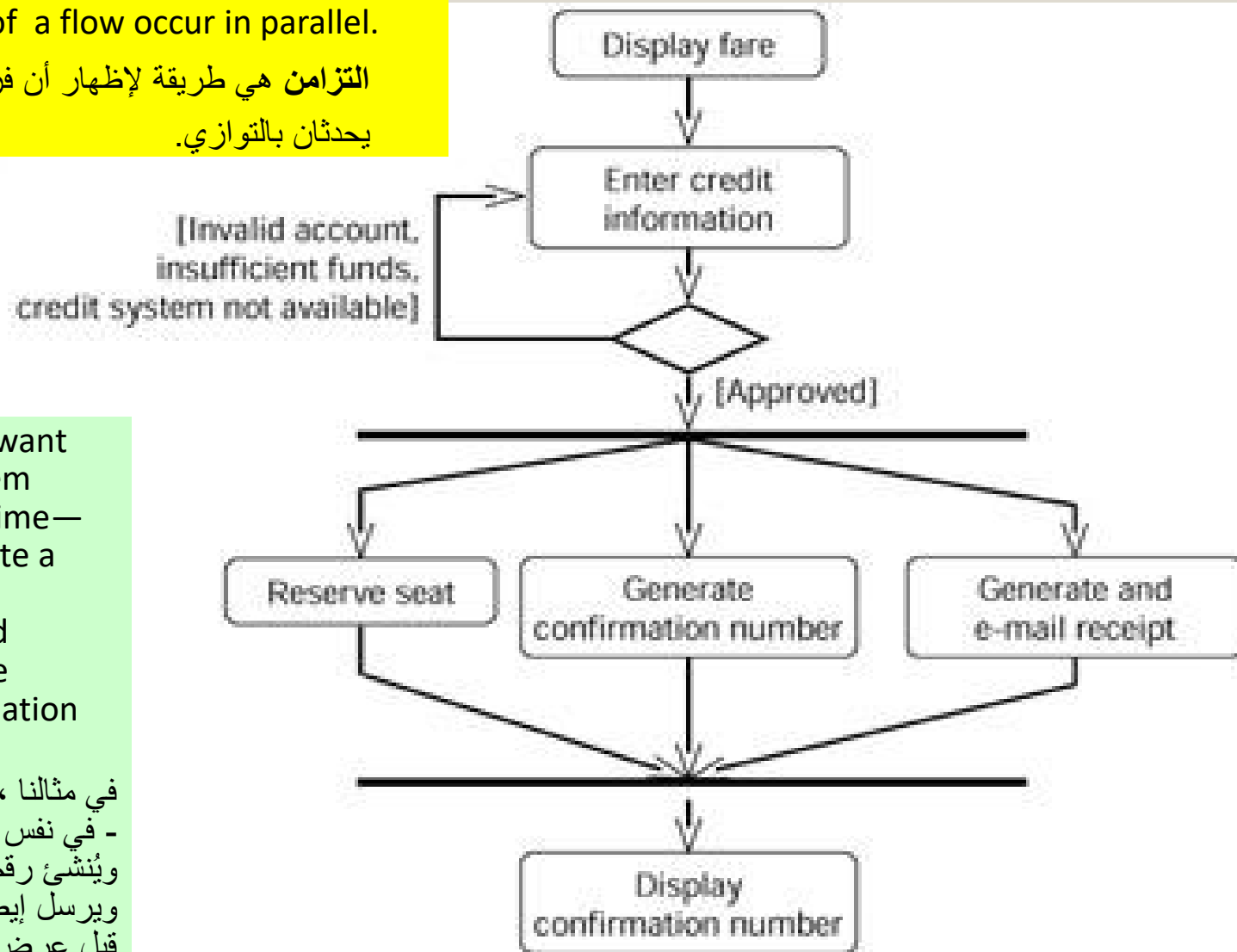
# Activity Diagrams-8

A **synchronization** is a way to show that two or more branches of a flow occur in parallel.

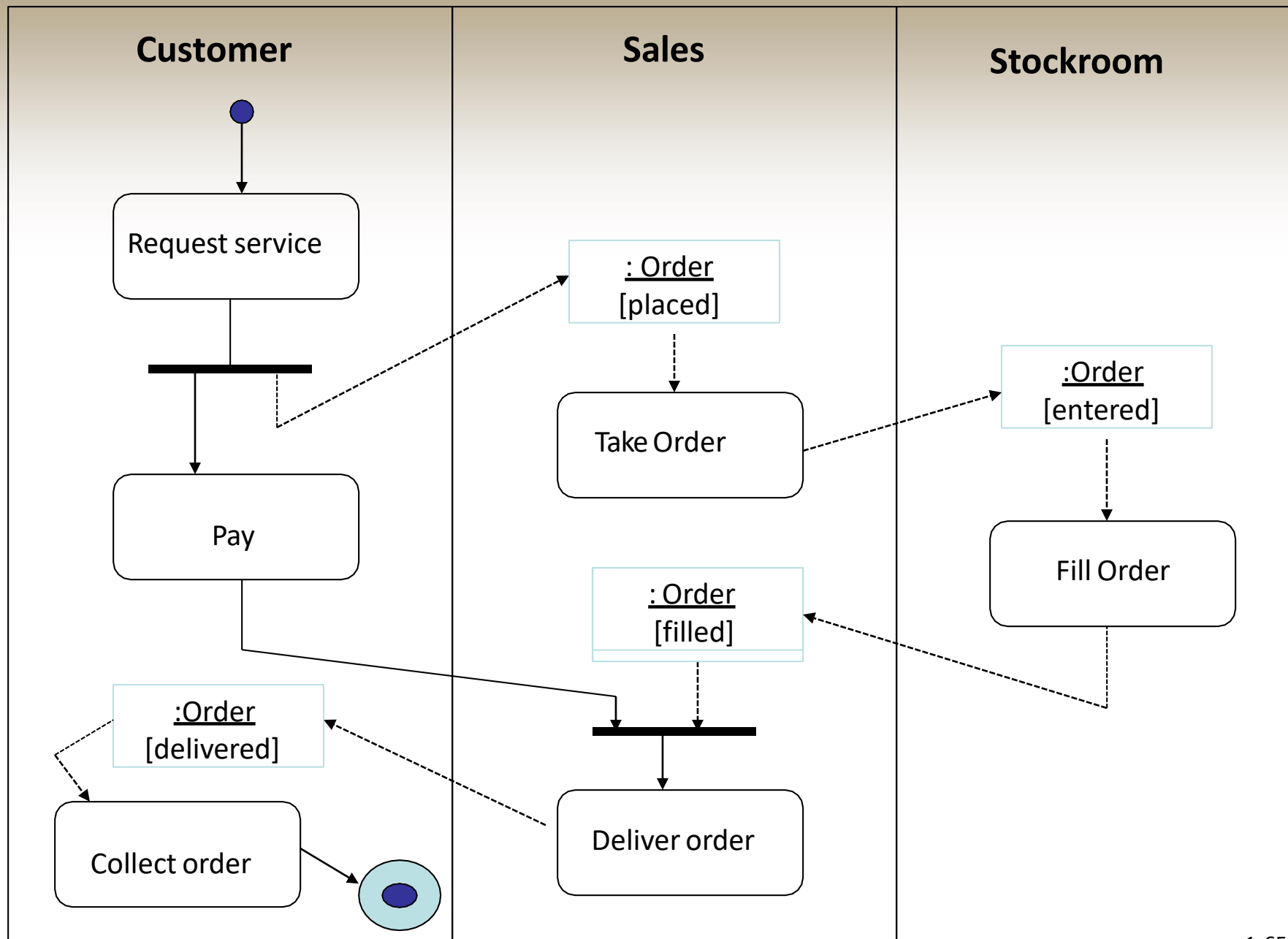
التزامن هي طريقة لإظهار أن فرعين أو أكثر من التدفق يحدثان بالتوازي.

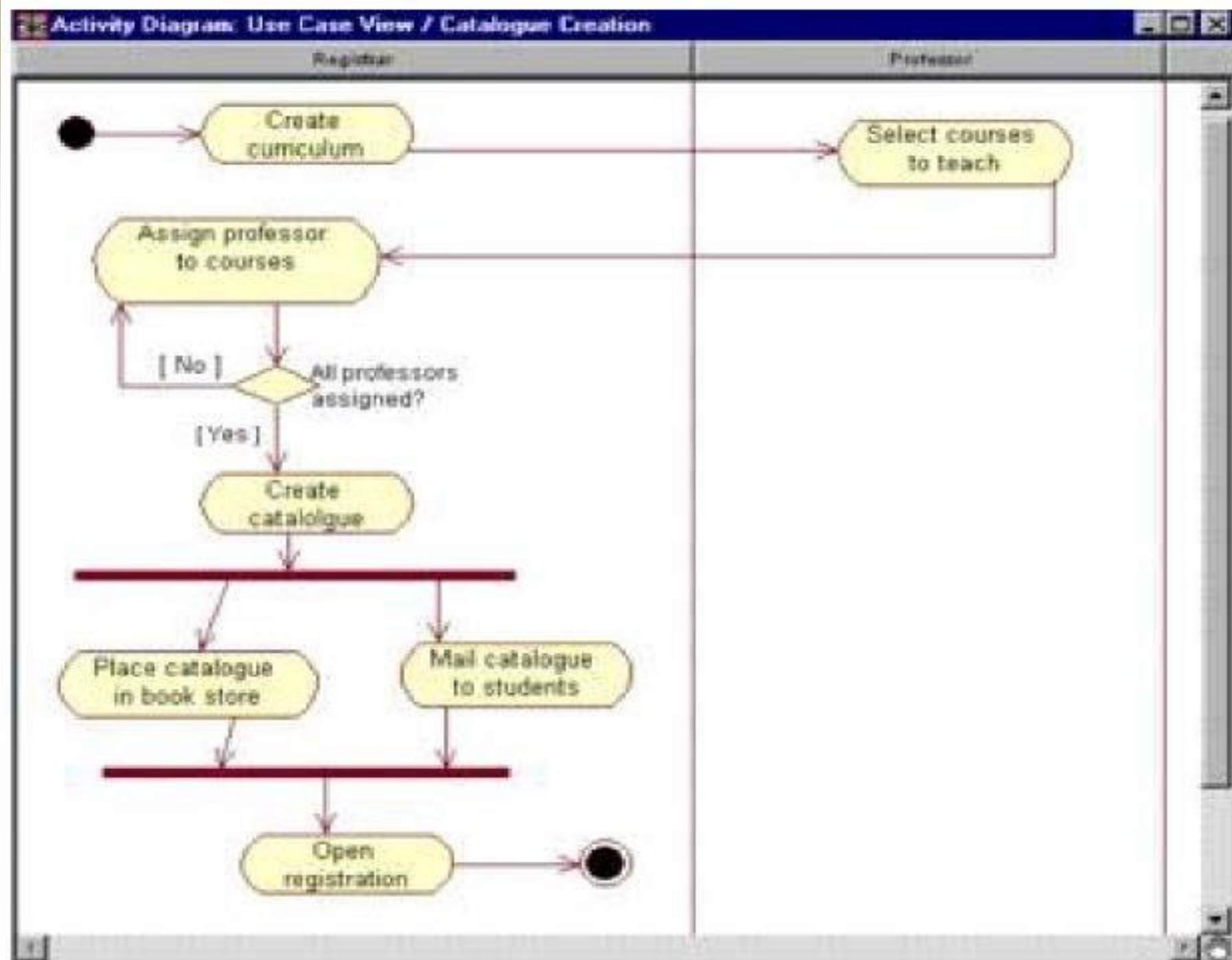
In our example, if we want to show that the system would—at the same time—reserve a seat, generate a confirmation number, generate a receipt, and e-mail a receipt before displaying the confirmation number,

في مثالنا ، إذا أردنا إظهار أن النظام - في نفس الوقت - سيحجز مقعدًا ، ويُنشئ رقم تأكيد ، ويُنشئ إيصالًا ، ويرسل إيصالًا بالبريد الإلكتروني قبل عرض رقم التأكيد ،











# Chapter 4,5: Use-Case Analysis (Object Interaction)

Chapter 4: Finding Classes

Chapter 5: Discovering Object Interaction

Source: Mastering UML with Rational Rose 2002

OOAD Using the UML - Use-Case Analysis, v 4.2

Khalil Barhoum, Isa University

Email: [kbarhoum@ipu.edu.jo](mailto:kbarhoum@ipu.edu.jo)

# Objectives: Use-Case Analysis

- ▶ Understand the purpose of Use-Case Analysis and where in the lifecycle it is performed
- ▶ Identify the classes needed to accommodate a use- case flow of events (Analysis Classes)
- ▶ Distribute the use-case behavior to those classes, identifying responsibilities of the classes
- ▶ Model analysis class interactions in interaction diagrams
- ▶ The analysis classes and the initial use-case realizations are the key model elements that we develop in this activity.

# Use Case Analysis

- ▶ The focus during Use-Case Analysis is on a **particular use case**.
- ▶ In Use-Case Analysis, we identify the analysis classes and define their responsibilities.
  - (At this time, we speak of '**responsibilities**.' (May even use the term, '**services**.')) Later, these might become 'methods' or 'member functions.' But that is a matter for implementation NOT initial design.)
- ▶ The allocation of responsibility is modeled in use-case realizations that describe how analysis classes collaborate to perform (realize) use cases.

# Realising a Use Case

- ▶ A Use Case Realisation is a group of classes whose methods and communication are sufficient to achieve the functionality of the Use Case
- ▶ Class Model
- ▶ Sequence Diagrams
- ▶ [Collaboration Diagrams]
- ▶ The full system can be implemented once individual Use Case realisations have been integrated to form one Class Model.
- ▶ RUP approaches this through one Use Case at a time, with refinements as necessary
- ▶ Each Use Case considered will refine/alter the 'big picture'



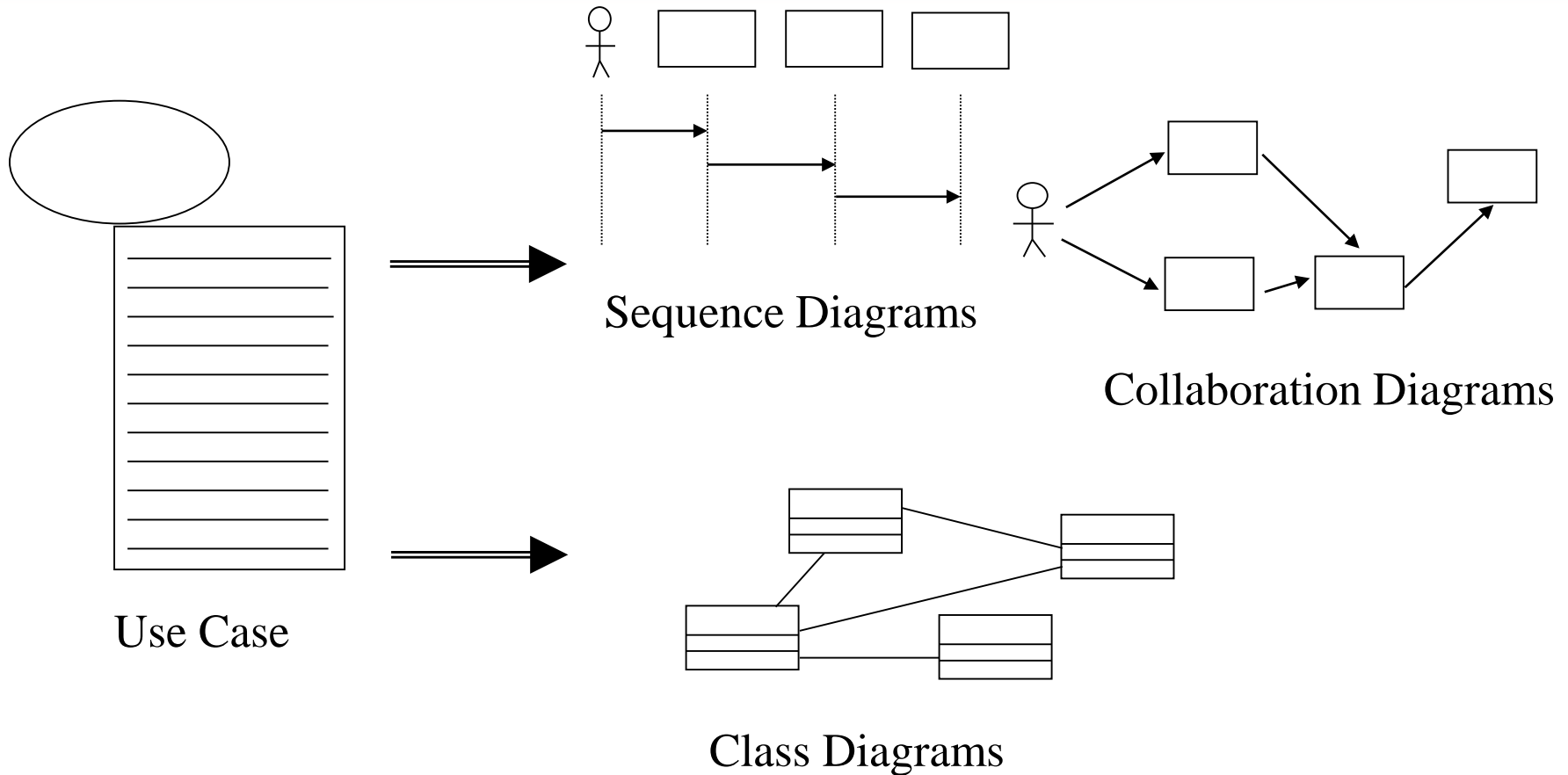
# Use-Case Analysis Steps – Major ones

## ► For each use-case realization, Do:

- Study Use Case flow of events - identify analysis classes from the Use Case narrative. These consist of interface, control, & data classes.
- Allocate behaviors (responsibilities) to these classes.
  - These are the little things the class must do....in some cases, these are merely options afforded to the user like add new customer() or delete customer ()....
- Distribute Use-Case Behaviors to Classes that you identify
- Describe Attributes (properties) of each analysis class.
- Show associations between the collaborating classes.

► Let's look at each of these two major items in particular....

# Realisation of a Use Case



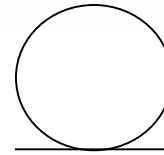


# Kinds of Analysis Classes

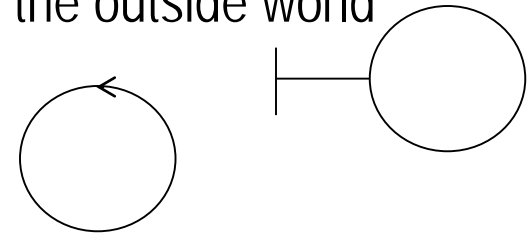
- ▶ The boundary between the system and its actors (interfaces... to end-users, external systems, devices...)
- ▶ The information a system uses (data), and
- ▶ The control logic of the system (who does what)
  
- ▶ So, we isolate the different kinds of concerns and use analysis classes to capture these responsibilities.
- ▶ Each category of analysis class has a typical set of duties & responsibilities
- ▶ The complete behavior of a use case must be distributed to analysis classes

# Analysis Classes

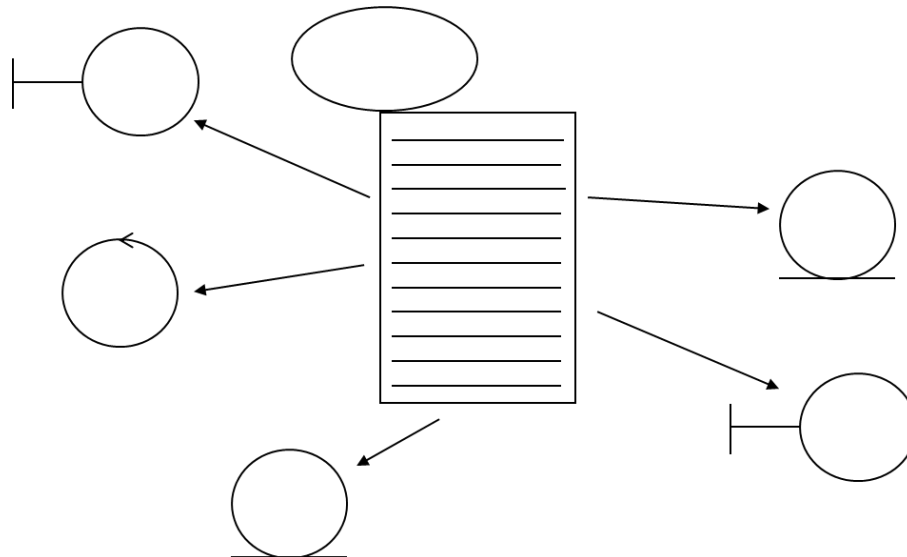
- ▶ Classes describing persistent data are not enough to describe the entire system behaviour
- ▶ During analysis, it is useful to include:
  - Entity objects (persistent data)



- Boundary objects (interface between the system and the outside world)



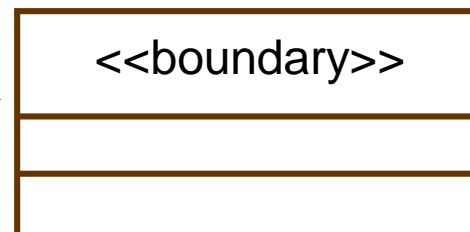
- Control objects (mediation between other objects)



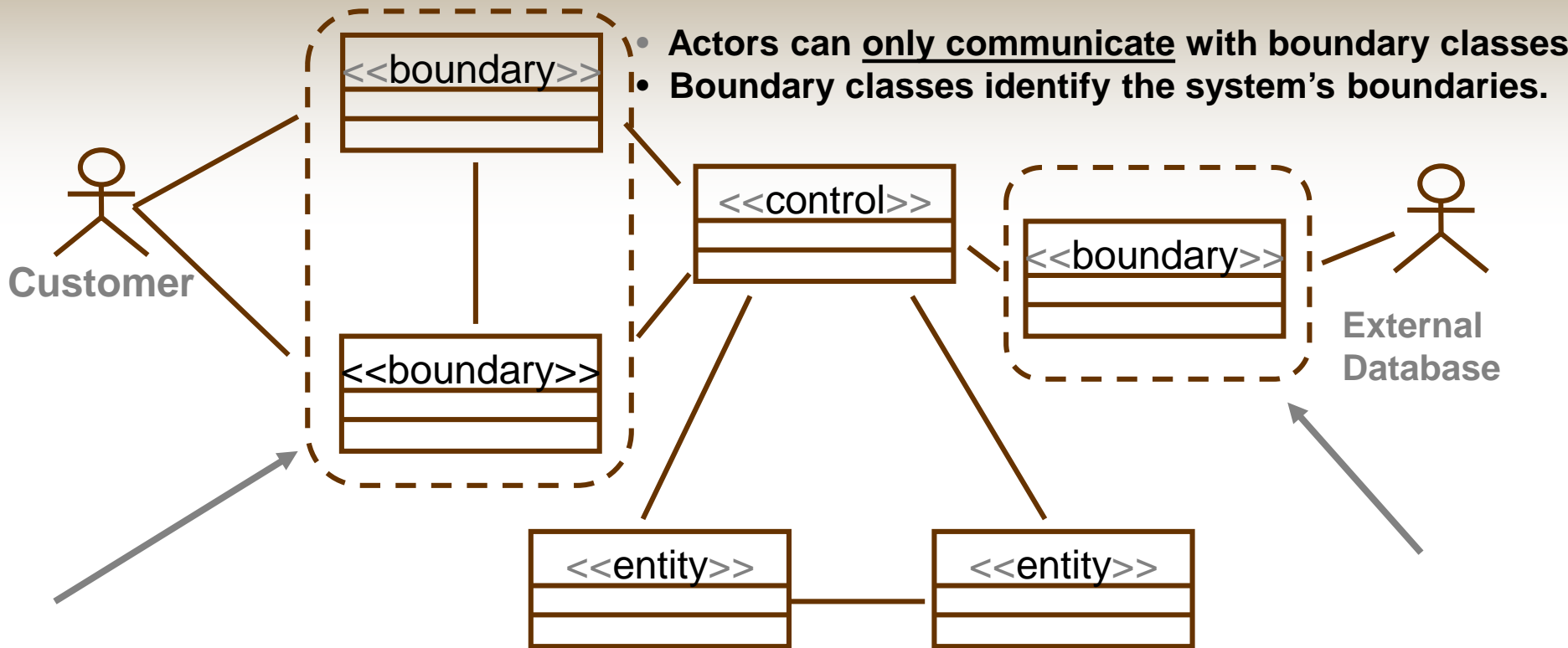
# What is a Boundary Class?

- ▶ Insulates the system from changes in the outside
- ▶ Three types of Boundary Classes
  - 1. User interface classes – classes that facilitate communication with human users of the system
    - ▶ Menus, forms, windows, etc. User interface classes....
  - 2. System interface classes – classes which facilitate communications with other systems. Very Important!
    - ▶ These boundary classes are responsible for managing the dialogue with the **external system**, like getting data from an existing database system or flat file...
    - ▶ Provide an interface to that system (like a VSAM file)
  - 3. Device Interface Classes – provide an interface to devices which detect external events – like a sensor or ... ➔ One boundary class per use case/actor pair

*Analysis class  
stereotype*



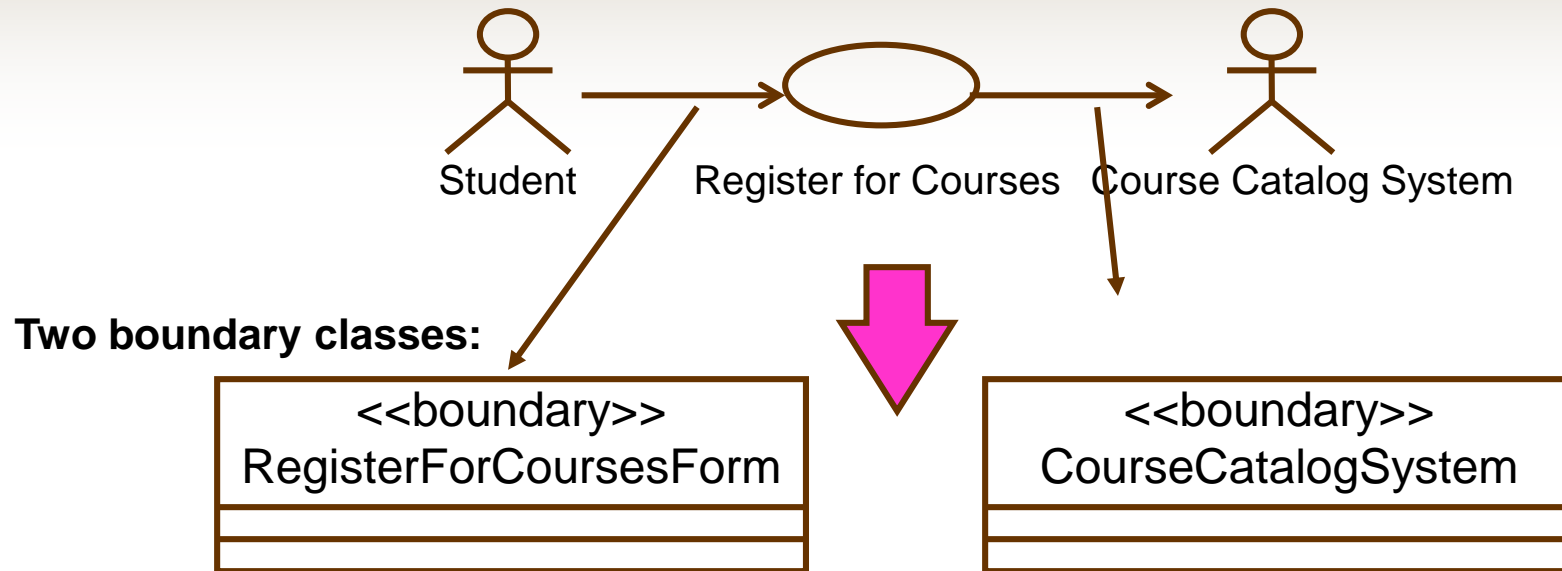
# The Role of a Boundary Class



- Boundary class - used to model interaction between system's surroundings and its inner workings.
- Boundary Classes model parts of the system that **depend** on its surroundings.
- Entity and control classes model parts that are **independent** of system's surroundings.
- Examples of boundary classes: Classes that handle GUI or communications protocols.

# Example: Finding Boundary Classes

- One boundary class per actor/use case pair:



- The RegisterForCoursesForm contains a Student's "schedule-in-progress". It **interfaces** with the actor and displays a list of Course Offerings for the current semester from which the Student may select specific courses to be added to his/her Schedule. (**Note that this description comes directly from the use case specification. We capture that behavior and encapsulate it (for now) into a boundary class with this description.**)
- The CourseCatalogSystem **interfaces** with the legacy system that provides the unabridged catalog of all courses offered by the university.

# Boundary Class – As a User Interface classes

- ▶ 1. Concentrate on what information is presented to the user
  - Do NOT concentrate on the UI details (windows, forms...)
  - Analysis Classes are meant to be a first cut at the abstraction of the system.
  - The boundary classes may be used as 'holding places' for GUI classes. (Addressed in more detail later in design)
  - ➔ Do not do a GUI design in analysis, but isolate all environment-dependent behavior. (Likely you may be able to reverse engineer a GUI component and tailor it.)
  - If prototyping the interface has been done, these screen dumps or sketches may be associated with a boundary class.
  - Only model the key abstractions of the system – not every button, list, etc. in the GUI.

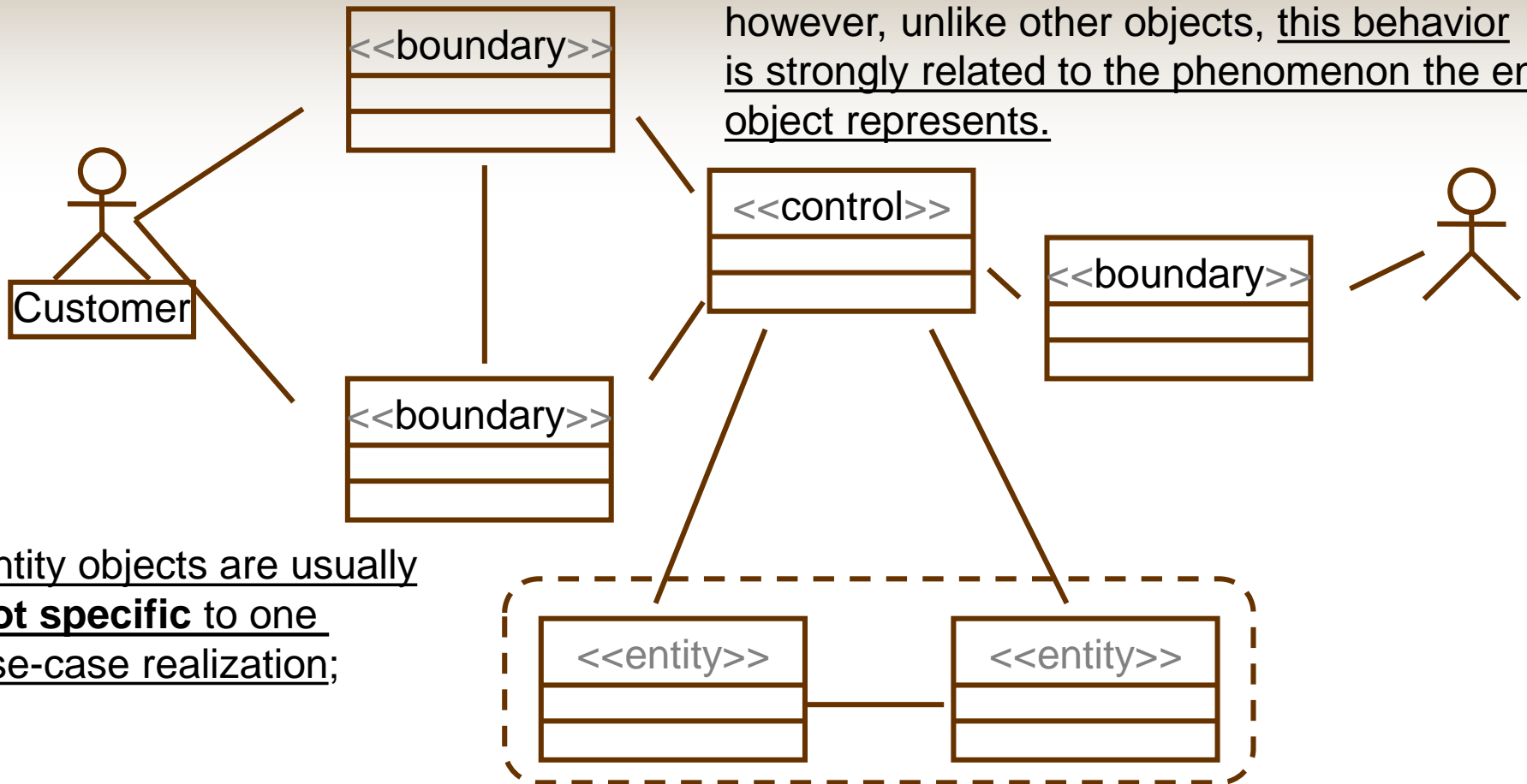


# What is an Entity Class? (recall: **boundary**, entity, **control...**)

- ▶ These are objects that hold information. They may eventually map to some of the tables and fields in the database. Many of the nouns in the flow of events will give you entity objects.
- ▶ Represent stores of information in the system
- ▶ Used to represent the key concepts the system manages. (Core Abstractions)
- ▶ The main responsibilities of entity classes are to store and manage information in the system.
- ▶ Often **persistent**; may well come from domain model (business entities)

# The Role of an Entity Class

Entity objects can have **complicated behavior**; however, unlike other objects, this behavior is strongly related to the phenomenon the entity object represents.



Entity objects are usually **not specific** to one use-case realization;

The values of an entity object and their relationships are often given by an actor. Entity objects are **independent** of the environment (actors)

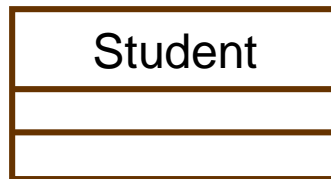


# Example: Finding Entity Classes

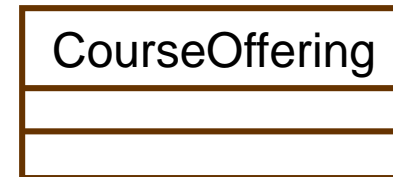
- ▶ ➔ Use use-case flow of events, the domain model, and glossary as inputs.
  - The more of these you have the better you are!
- ▶ Traditional, filtering nouns approach (but I'd recommend using classes from the domain model (business entities) as a starting place, if available)
  - Underline noun clauses in the use-case flow of events
  - Remove redundant candidates; Lots of synonyms!
  - Remove vague candidates
  - Remove actors (out of scope)
  - Remove implementation constructs
  - Remove attributes (save for later)
    - ▶ Lots of times, candidate '**nouns**' may become '**attributes**' in classes.
  - Remove operations

# Example: Candidate Entity Classes

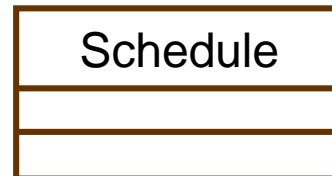
- ▶ Register for Courses Use Case
  - Specific scenario? Likely, Create Schedule.



A person enrolled in classes at the university



A specific offering for a course including days of week and times



The courses a student has selected for current semester

**Note:** these are candidate entity classes only. So far, they do not possess responsibilities, attributes, associations, etc...

# Candidate Entity Classes – Actors – same name?

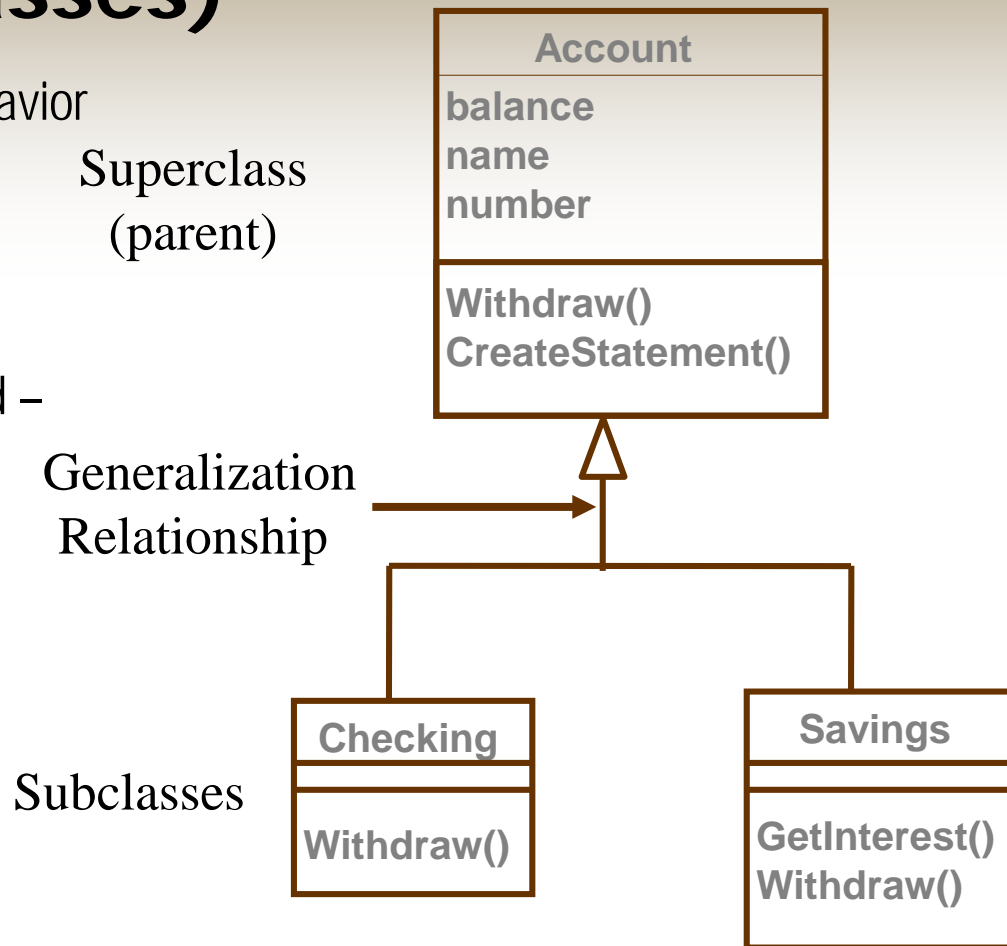
- ▶ Sometimes there's a need to model **information** within the system. This is not the same as modeling the **actor** (actors are external. by definition) with the same name.
- ▶ A course registration system maintains information about the Student object which is independent of the fact that the Student also plays a role as an Actor of the system.
  - Completely independent of each other
  - **Student class** (entity) will exist whether or not the student is an actor to the system.

# Review: Generalization (in entity classes)

- ▶ One class shares the structure and/or behavior of one or more classes
- ▶ "Is-a-kind of" relationship
- ▶ In analysis, use sparingly
- ▶ Inheritance relationships may be identified – especially in entity classes.

In analysis, generalization should be used to model shared **behavioral semantics** only (that is, generalization that passes "is a kind of" test)

The use of **generalization** makes the definition of the **abstractions** easier to document and understand.



When generalization is found, create a common super-class that contains common attributes, associations, aggregations, and operations

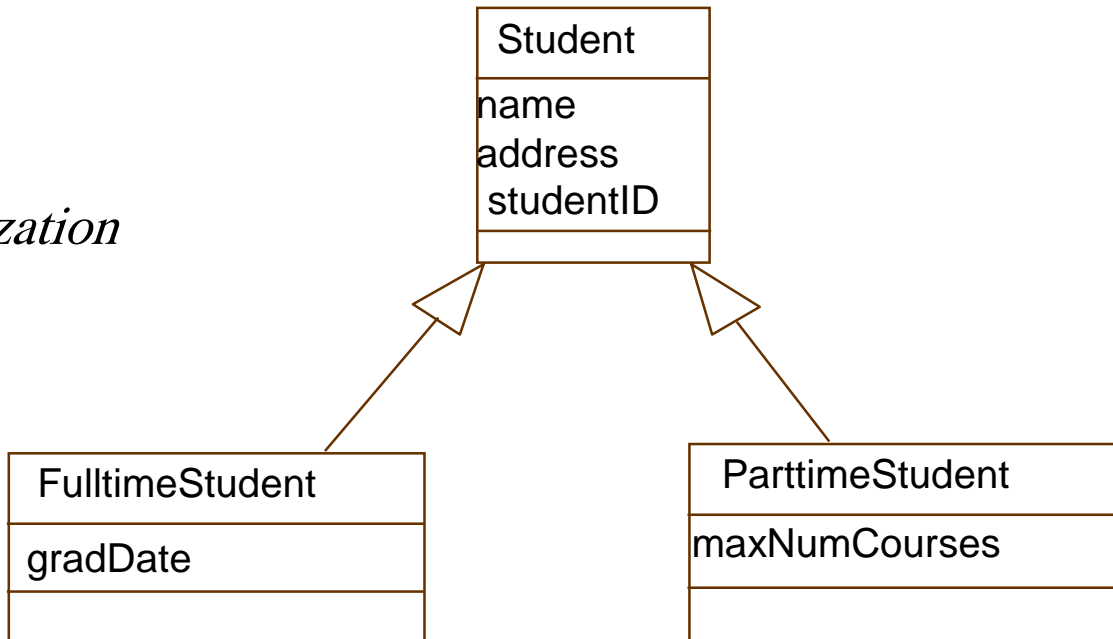
# Example: Generalization (Shared Semantics)

*Without  
Generalization*

Part-timeStudent
name
address
studentID
numberCourses

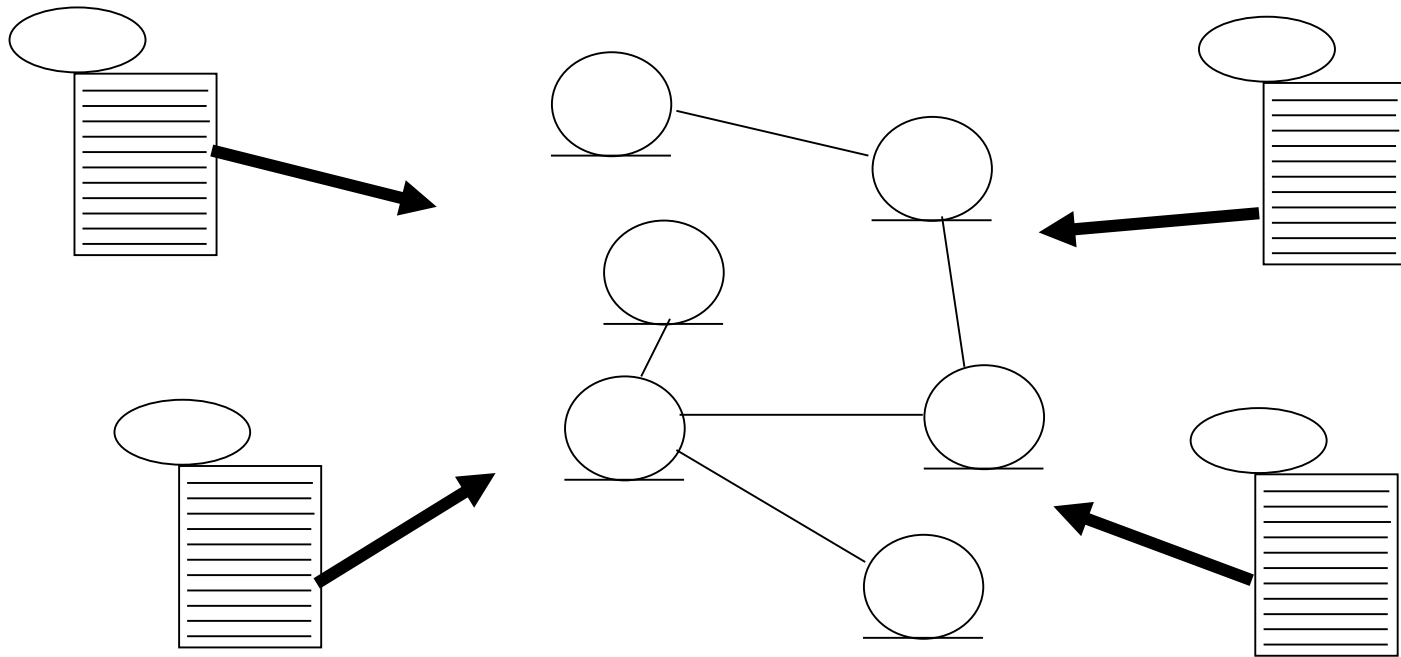
Full-timeStudent
name
address
studentID
gradDate

*With  
Generalization*



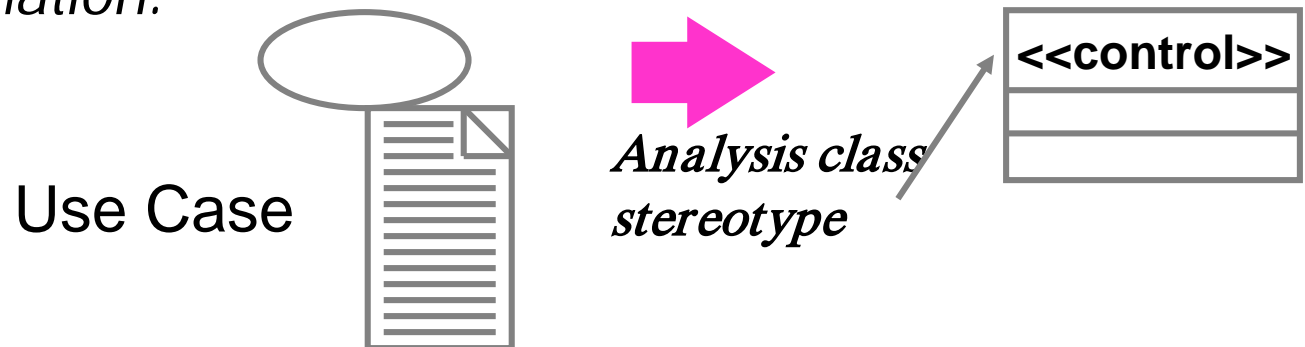
# Integrated Class Model

*Integrate Entity Classes for database design*



# What is Control Class? (recall: boundary, entity, control...)

- ▶ Is a Use-case behavior coordinator;
  - sequences; controls; orchestrates use-case.
  - Might 'say' control objects "run" the use-case realizations.
- ▶ *One control class per use case (generally)*
- ▶ *The system sometimes can perform some use cases without control classes (just using entity and boundary classes) – particularly use case only involves simple manipulation of stored information.*



*Use-case dependent, Environment independent*

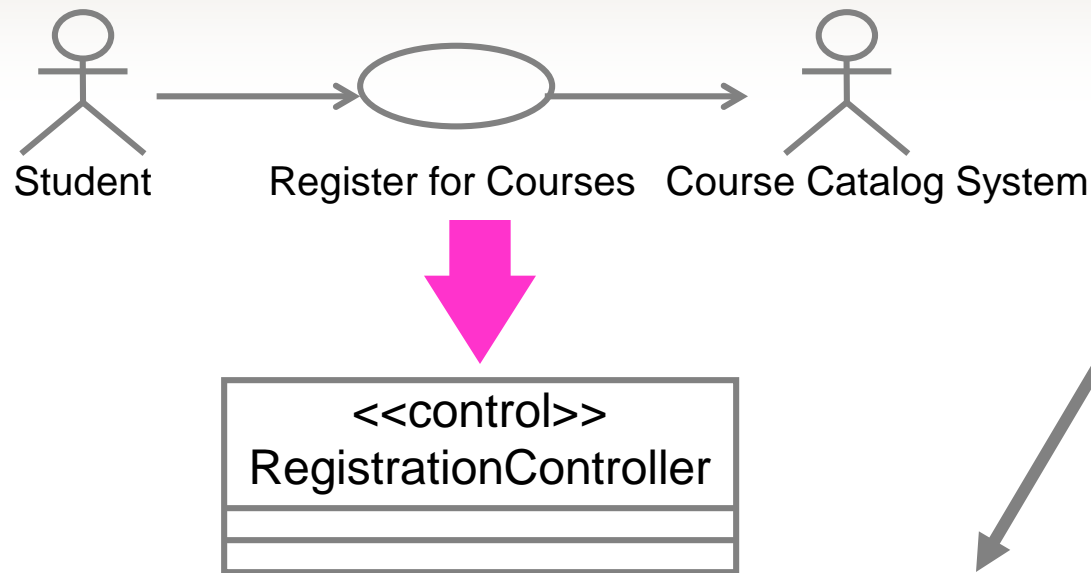
# Role of Control Class and Control Objects

- ▶ Control classes contribute to understanding the system.
  - **Represents** the **dynamics** of the system,
  - Handles the **main tasks and control flows.**
- ▶ When the system performs the use case, a control object is **created**.
- ▶ Control Objects usually **die** when their corresponding use case has been performed.
  - (Normally NOT persistent)



# Example: Finding Control Classes

- Recommend: Identify one control class per use case (again)



➔ Each control class is responsible for **orchestrating/controlling** the processing that **implements** the functionality described in the associated use case.

Here, the RegistrationController <<control>> class has been defined to orchestrate the Register for Courses processing (**sequencing of activities**) within the system.

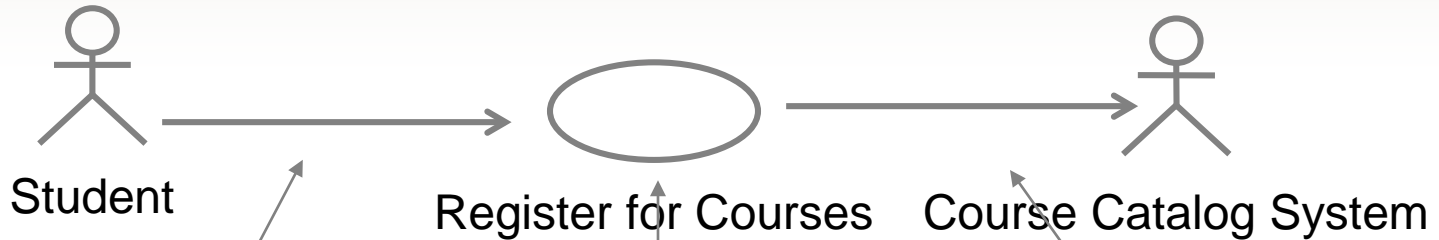
➔ (Controller accepts inputs, '**knows**' where required data and functionality reside, sends key messages to entities, sequences all actions to satisfy use case, sends data back to input actor, etc....)

# Now, Summarizing Analysis Classes in general:

- ▶ Summary of Analysis Classes: View of Participating Classes (VOPC)
- ▶ For **each use-case realization** there is one or more class diagrams depicting its participating classes, along with their relationships.
- ▶ Such class diagrams have been called “View of Participating Classes” diagrams (**VOPC**, for short) – next overhead...

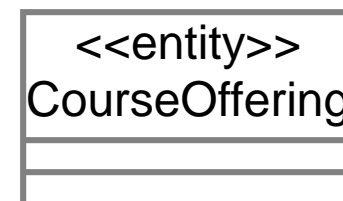
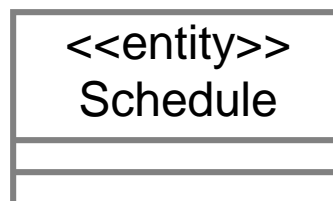
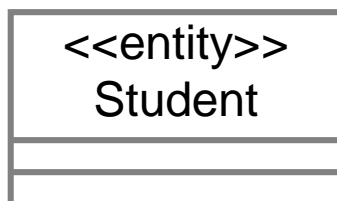
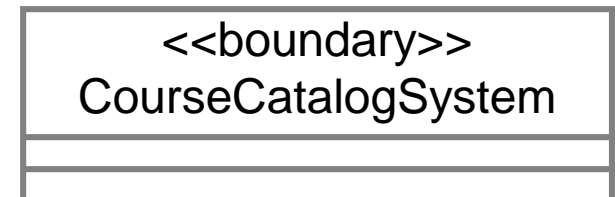
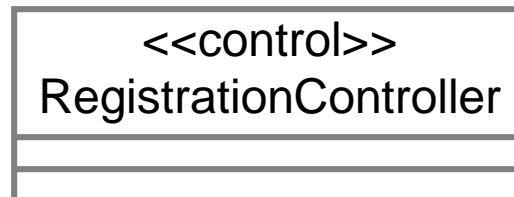
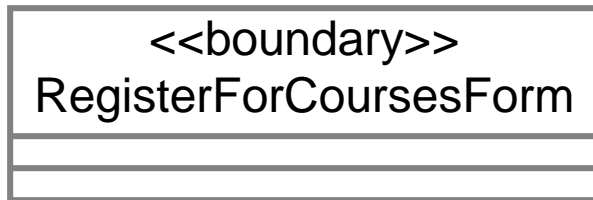
# Example: Summary: Analysis Classes - VOPC

- The diagram shows the **classes participating** in the Register for Courses use case
- The part-time student and full-time student classes have been omitted for brevity (they both inherit from Student. Class relationships will be discussed later.



## Use-Case Diagram

## Analysis Model (classes only listed – no relationships shown here...)



# Use-Case Analysis Steps – Next Major Step...

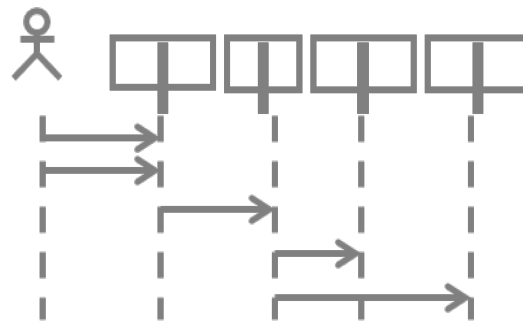
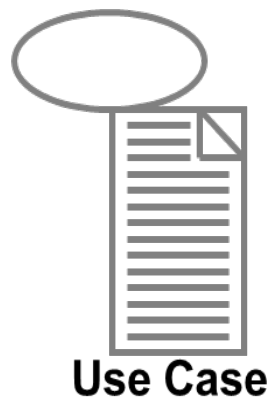
- ▶ Supplement the Use-Case Descriptions
- ▶ For each use-case realization
  - Find Classes from Use-Case Behavior - DONE! But nothing in them!
  - ★ **→ Distribute Use-Case Behaviors to these Classes**

Now we need to **allocate** responsibilities of the use cases to the analysis classes and **model** this allocation by describing the way the class instances collaborate to perform the use case in use-case realizations.

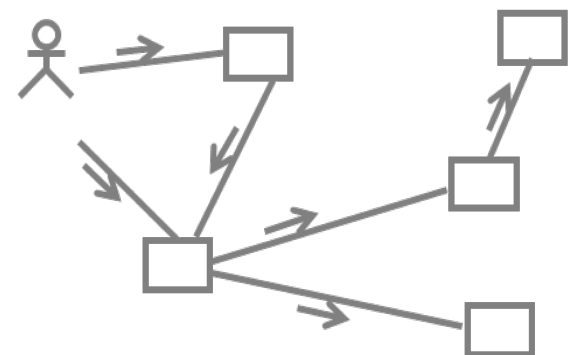
---

Purpose of Distributing Use Case Behavior to Classes is to:

- Express the use-case behavior in terms of collaborating objects, and **thus**
- Equivalently: **Determine** the responsibilities of analysis classes.



Sequence Diagrams



Collaboration Diagrams

# Guidelines: Allocating Responsibilities to Classes

- ▶ This allocation of responsibilities is crucial!
- ▶ Use analysis class stereotypes as a guide
  - Boundary Classes (the Interface)
    - ▶ Behaviors that involves communication with an actor
  - Entity Classes (Persistent Data)
    - ▶ Behaviors that involves the data encapsulated within the abstractions. (All data manipulation, retrieval...)
  - Control Classes (the Use Case flow of events)
    - ▶ Behaviors specific to a use case or part of a very important flow of events
- ▶ Notice, all these allocations are of behaviors.

*(continued)*

# Guidelines: Allocating Responsibilities to Classes (cont.)

- ▶ A driving influence on where a responsibility should go is the location of the data needed to perform the operation.
- ▶ Who has the data needed to perform the responsibility?
  - Example: "System displays Patient data on monitor."
    - ▶ Where is the patient data? **Patient** object. (entity object)
    - ▶ Who can 'get' the data? **Patient object!** (may imply a retrieval from a database, but ultimately the patient object has the data and the **responsibility** to get it.)
  - Who coordinates (issues message to patient object) to **get** this data?  
**Control class.**
  - Once the data is 'obtained,' who will "**display**" the data to the actor?  
**Boundary class. Look for data and verbs and nouns!**
  - For a class that has the data, put the responsibility for access, for manipulation, for modification, etc. with the data – best case!!!

# Guidelines: Allocating Responsibilities to Classes (cont.)

- ▶ Many authors feel that the use of control classes results in behavior being separated from data.
- ▶ This can happen if your control classes are not chosen wisely.
- ▶ If a control class is doing more than sequencing, then it is doing too much!
  - **For example**, in the Course Registration System, a student selects course offerings and if the course offering is available, the student is added to it.
  - **Who knows how to add the student** —the control class or the course offering? The right answer is the course offering.
  - **The control class** knows when the student should be added; the **course offering** knows how to add the student.
  - **A bad control class** would not only know when to add the student but how to add the student.

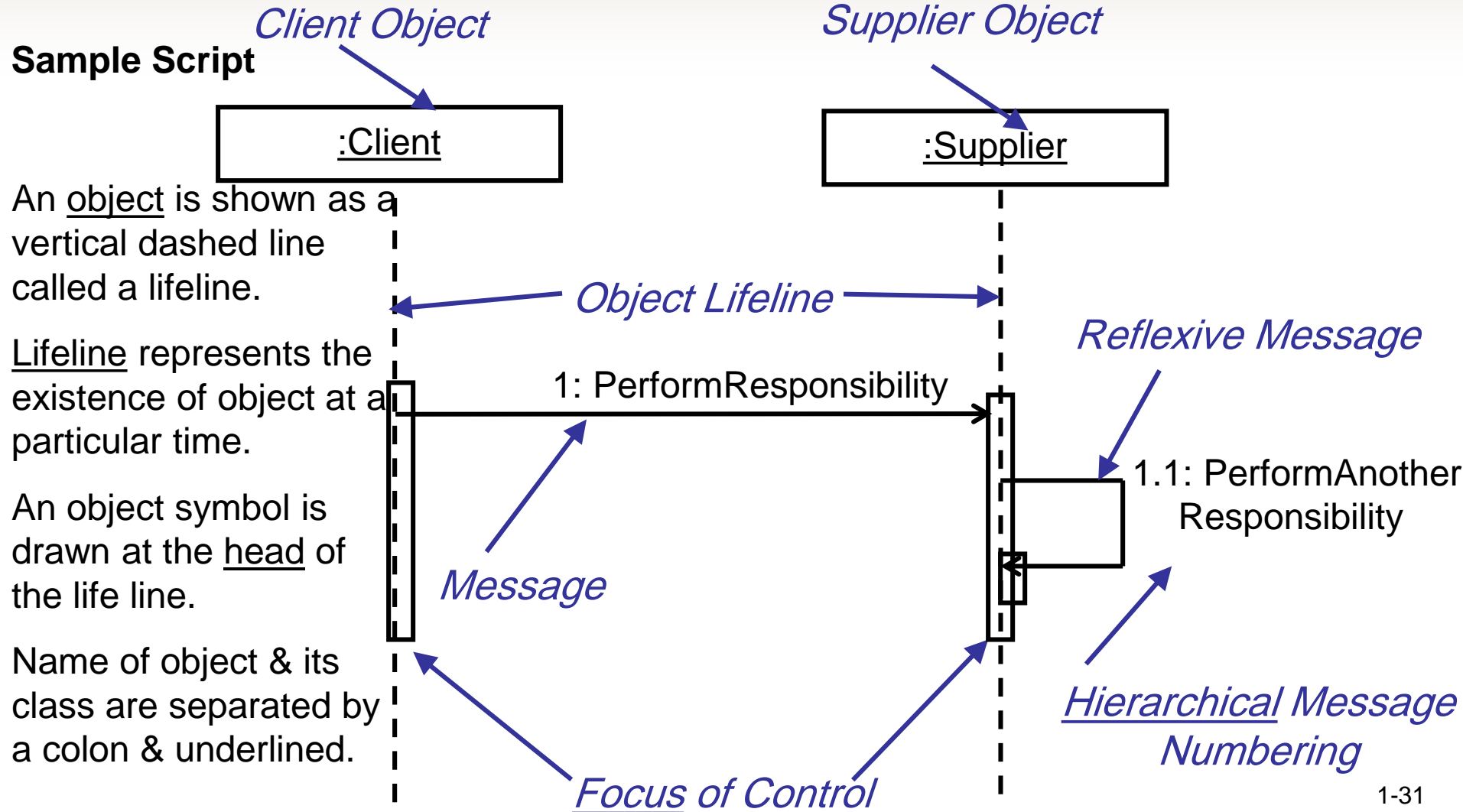
# Object Interaction Diagrams

- ▶ While the flow of events focuses on *what* the system needs to do, Sequence and Collaboration diagrams help to define *how* the system will do it.
- ▶ These diagrams focus on the objects that will be created to implement the functionality spelled out in the use cases.
- ▶ An *Interaction diagram* shows you, step-by-step, one of the flows through a use case
- ▶ An *Interaction diagram* shows the objects participating in a flow through a use case and the messages that are sent between the objects.
- ▶ While the flow of events focuses on what the system needs to do, Sequence and Collaboration diagrams help to define how the system will do it.

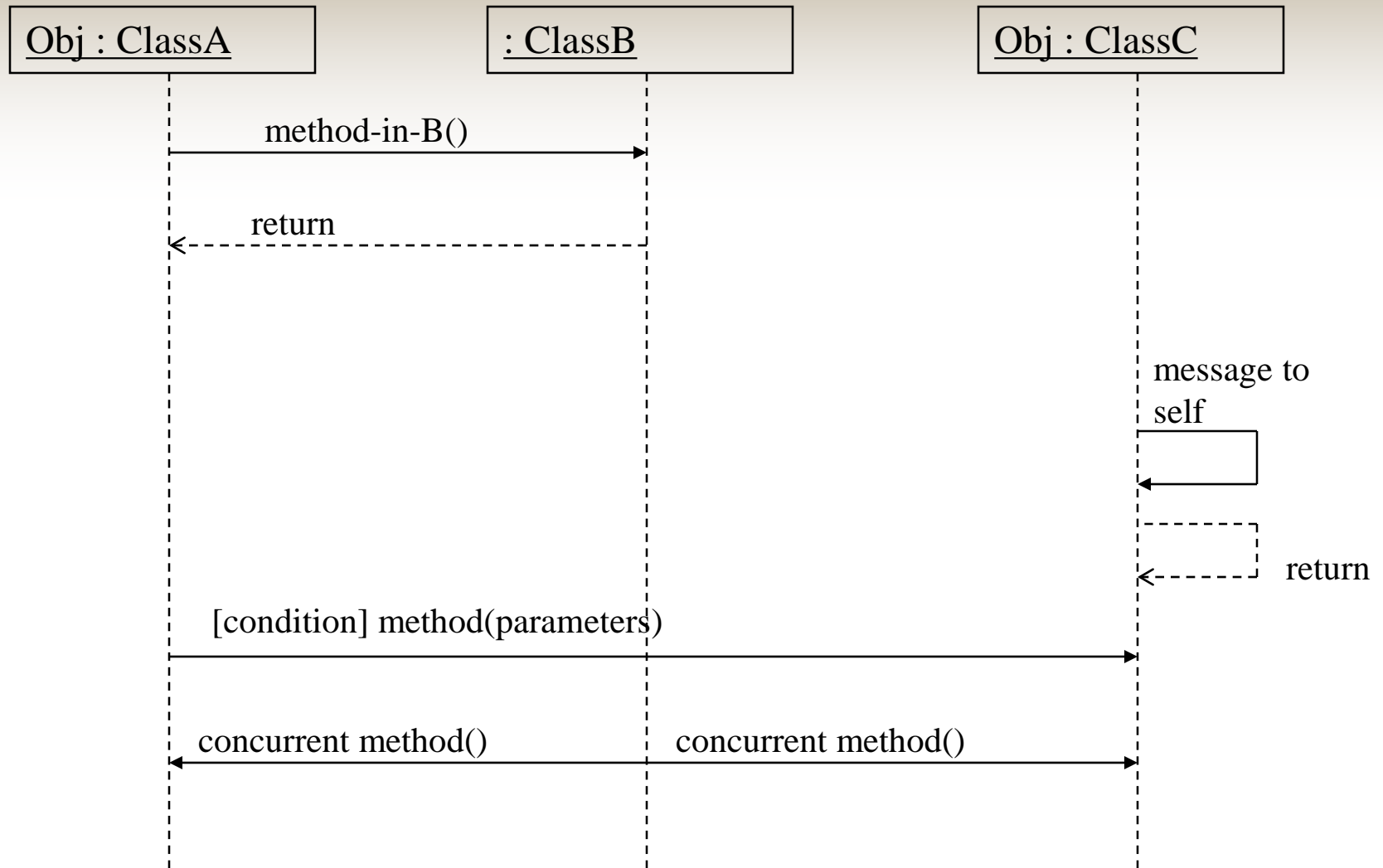


# The Anatomy of Sequence Diagrams

A Sequence Diagram describes a pattern of interaction among objects, arranged in a chronological order; it shows the objects participating in the interaction and the messages they send.



# Sequence diagram – basic syntax



# The Anatomy of Sequence Diagrams – the Message

- ▶ A **message** a one-way communication between objects
  - synonymous to “data flow”
  - may have parameters
  - generally corresponds to calling a method in the other object
  - may be **synchronous** (waiting for return) or **asynchronous** (continues without waiting for a response)
  - A message is shown as a horizontal solid arrow from the **lifeline** of one object to the **lifeline** of another object.
  - For a reflexive message, the arrow starts and finishes on the same lifeline.
  - The arrow is labeled with the name of the message, and its parameters.
  - The arrow may also be labeled with a sequence number.

# Sequence diagram - semantics

- ▶ The dotted vertical line indicates the life line of the corresponding object
- ▶ Solid arrows with solid arrowhead indicates messages
- ▶ Dashed arrows indicate return of control
  - these are optional; if provided, it indicates whether or not the corresponding message is synchronous or asynchronous
  - also indicates the scope of a message or stimulus execution and thus indicates grouping of messages or signals
- ▶ Messages may be augmented with conditions
  - these conditions may also indicate iterations of the method
- ▶ Messages can be concurrent
  - start from one object and invoke methods or send signals to more than one object at the same time

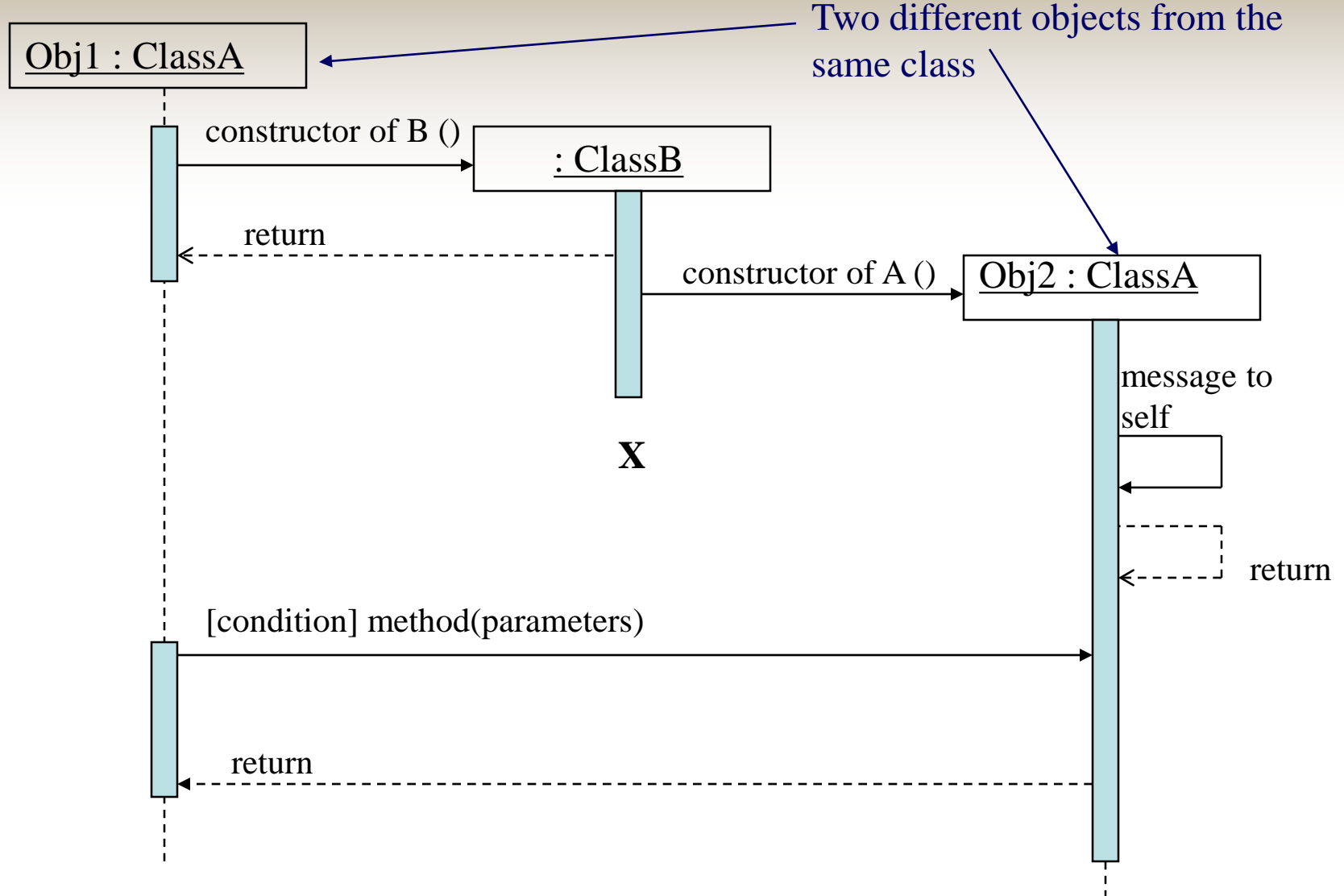
# Anatomy of Sequence Diagrams – Focus of Control

- ▶ **Focus of control** represents the relative time that the flow of control is focused in an object, thereby representing the time an object is directing messages.
- ▶ Focus of control is shown as narrow rectangles on object lifelines.
- ▶ **Hierarchical numbering** bases all messages on a dependent message.
- ▶ The dependent message is the message whose focus of control the other messages originate in.
  - For example, message 1.1 depends on message 1.
- ▶ **Scripts** describe the flow of events textually.
- ▶ Be certain to include the use-case text down the left margin of the sequence diagram (**style!**)

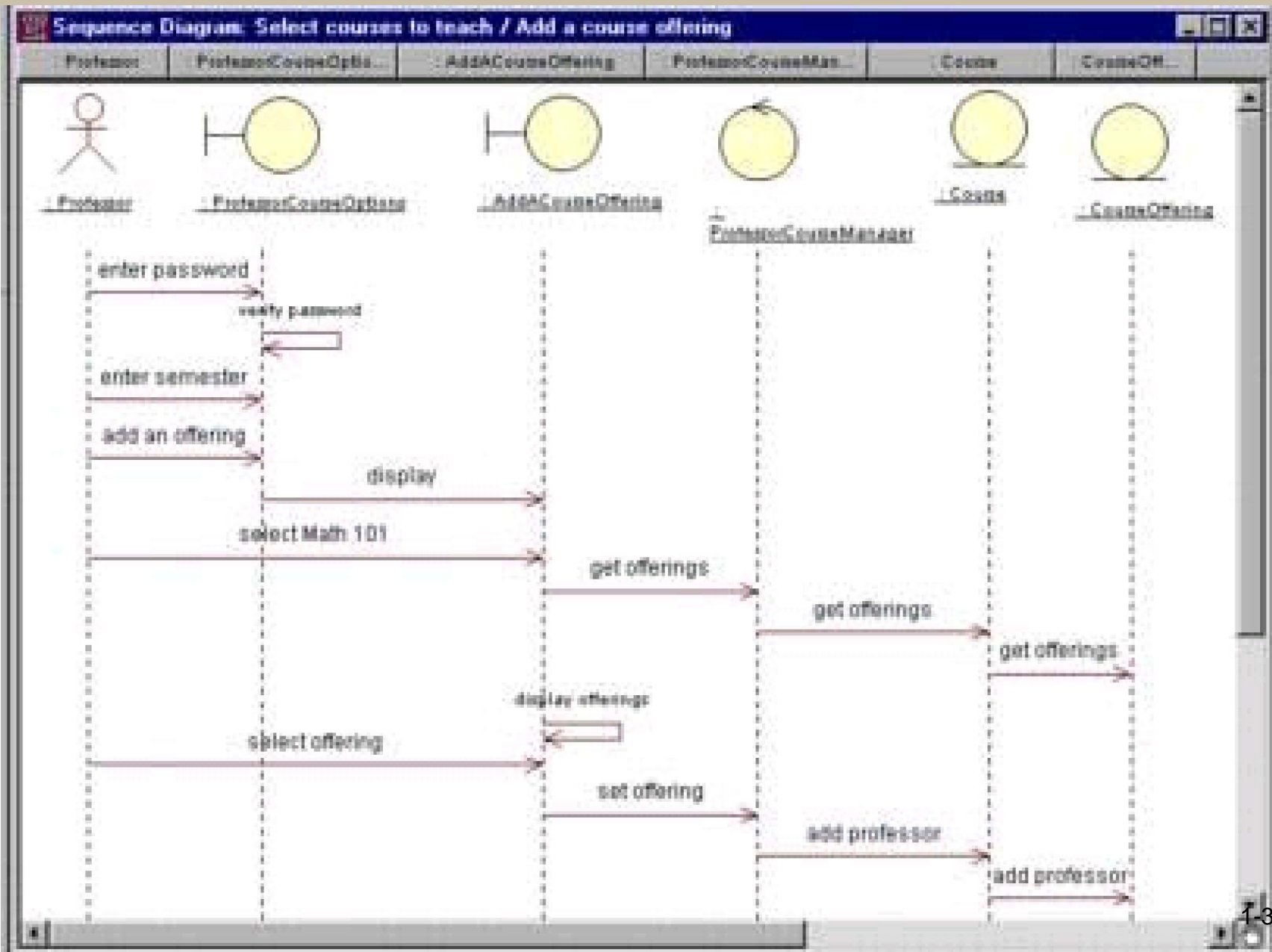
# Sequence diagram – semantics (continued)

- ▶ objects with dashed vertical line for the entire diagram have lifeline for the entire scenario
  - these objects are assumed to be already created before this scenario starts and assumed to exist after the scenario ends
- ▶ objects with short life span within a scenario can be shown differently (see the next diagram)
- ▶ there is no ordering among the placement of objects on the horizontal line
  - a designer may choose the ordering for the convenience of drawing the diagram
- ▶ the thin rectangle on each lifeline is called “activation” bar
  - it indicates the duration in which the object is active
  - the dotted line still indicates that the object is alive but not active
- ▶ the creation of an object is shown at the point of creation
  - see the creation of objects from class B and class A
- ▶ the termination of an object can be shown by placing an “X” at the bottom of its life line

# Sequence diagram – extended syntax



# Sequence Diagram for the Add a Course Offering Scenario





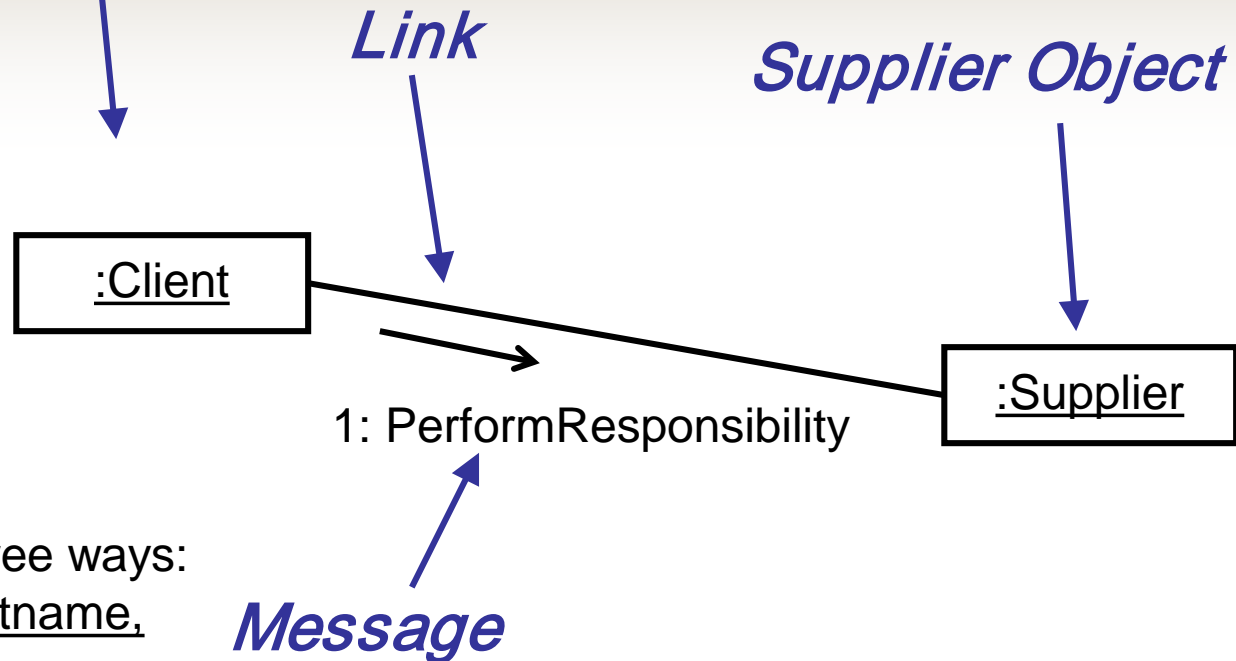
# Anatomy of Collaboration (Communications) Diagrams

*Client Object*

A **collaboration diagram** describes a **pattern of interaction** among objects.

It shows objects **participating** in the interaction by their **links** to each other and the **messages** that they send to each other.

(An **object** is represented in three ways: Objectname:Classname, Objectname, and :Classname)

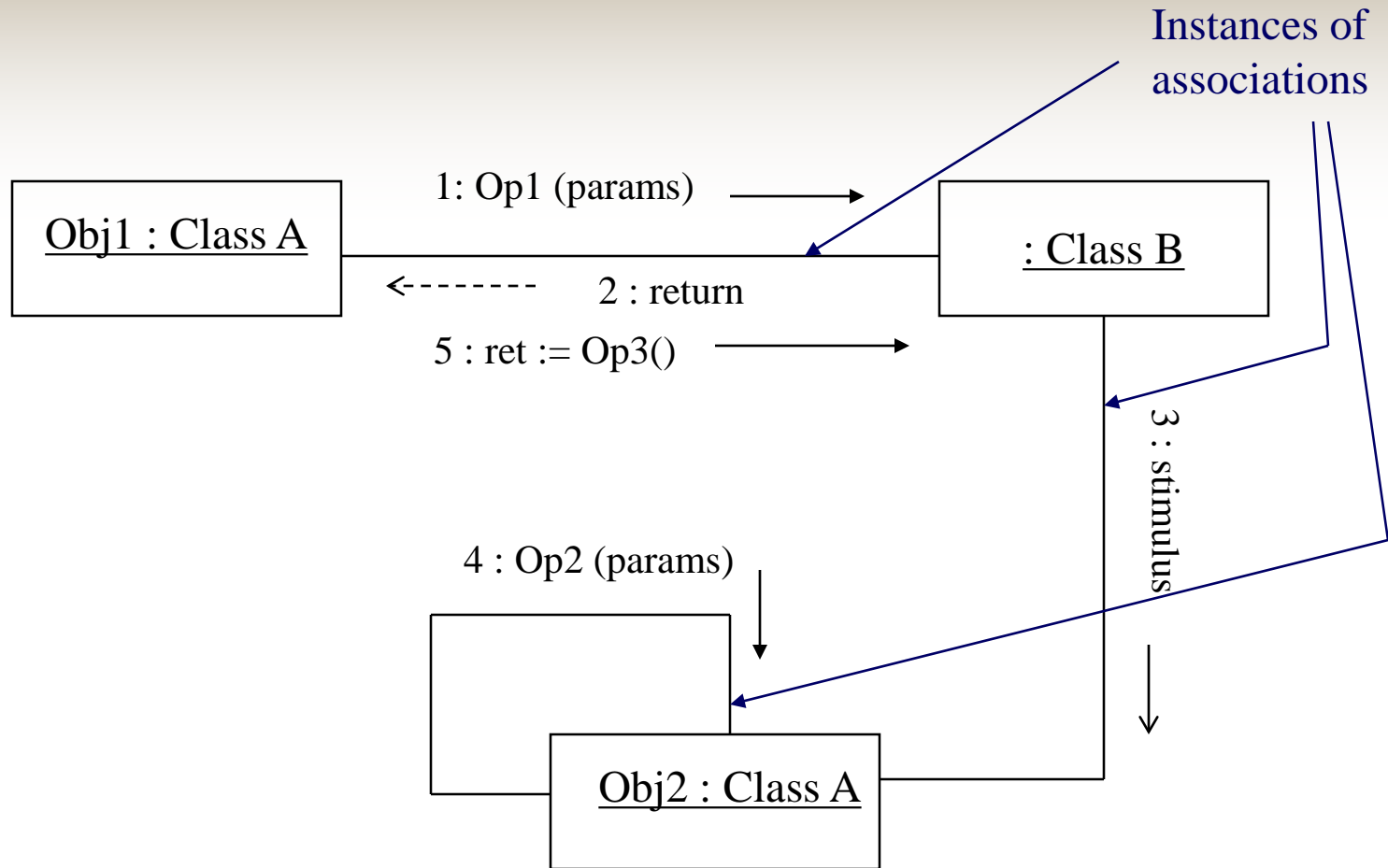


A **link** is a relationship among objects across which messages can be sent. In a collaboration diagram, a link is shown as a solid line between two objects. A link can be an instance of an association, or it can be anonymous – meaning that its association is unspecified.

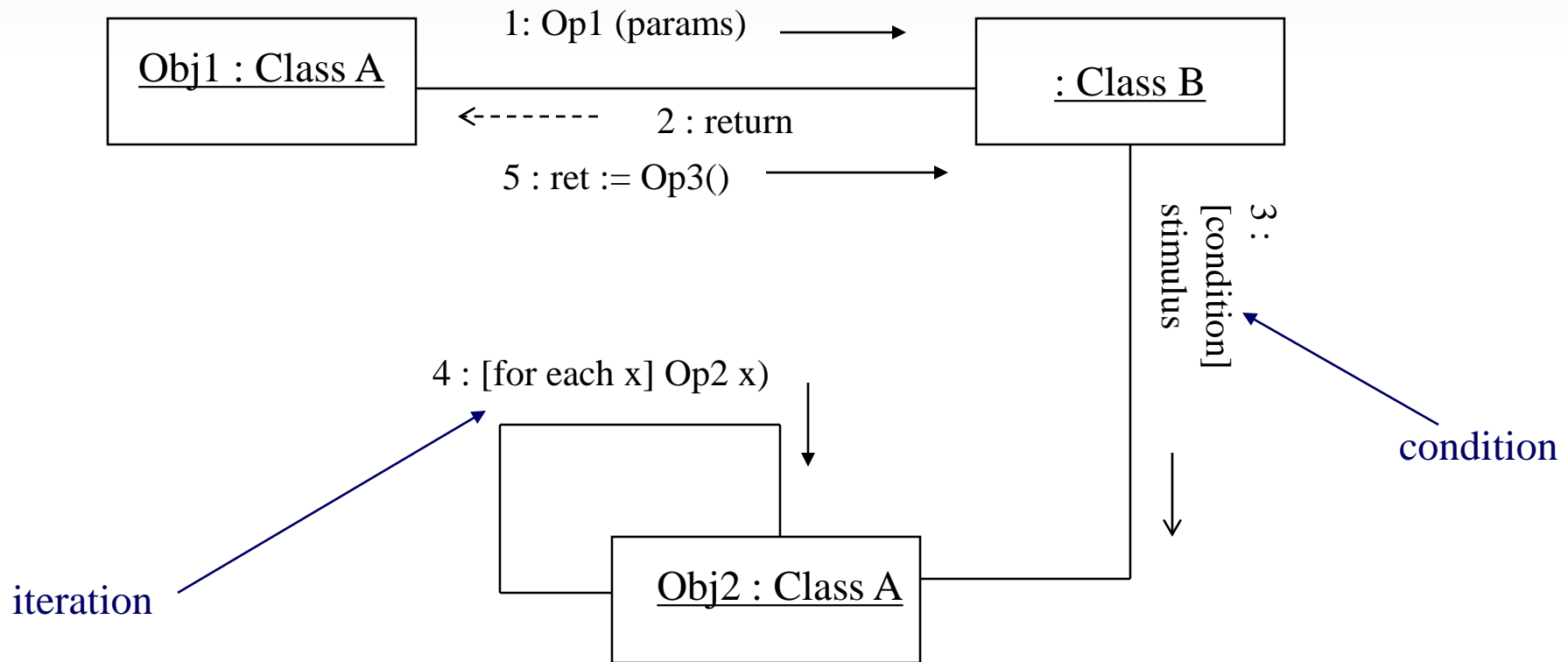
# Anatomy of Collaboration / Communications Diagrams - more

- ▶ A **message** is a communication between objects that conveys information with the expectation that activity will ensue. (back one to slide)
- ▶ In collaboration diagrams, a message is shown as a labeled arrow placed near a link.
  - This means that the link is used to transport, or otherwise implement the delivery of the message to the target object.
- ▶ The arrow points along the link in the direction of the target object (the one that receives the message).
- ▶ The arrow is labeled with the name of the message, and its parameters (no parameters shown here).

# Collaboration diagram – basic syntax



# Conditions and iterations - syntax



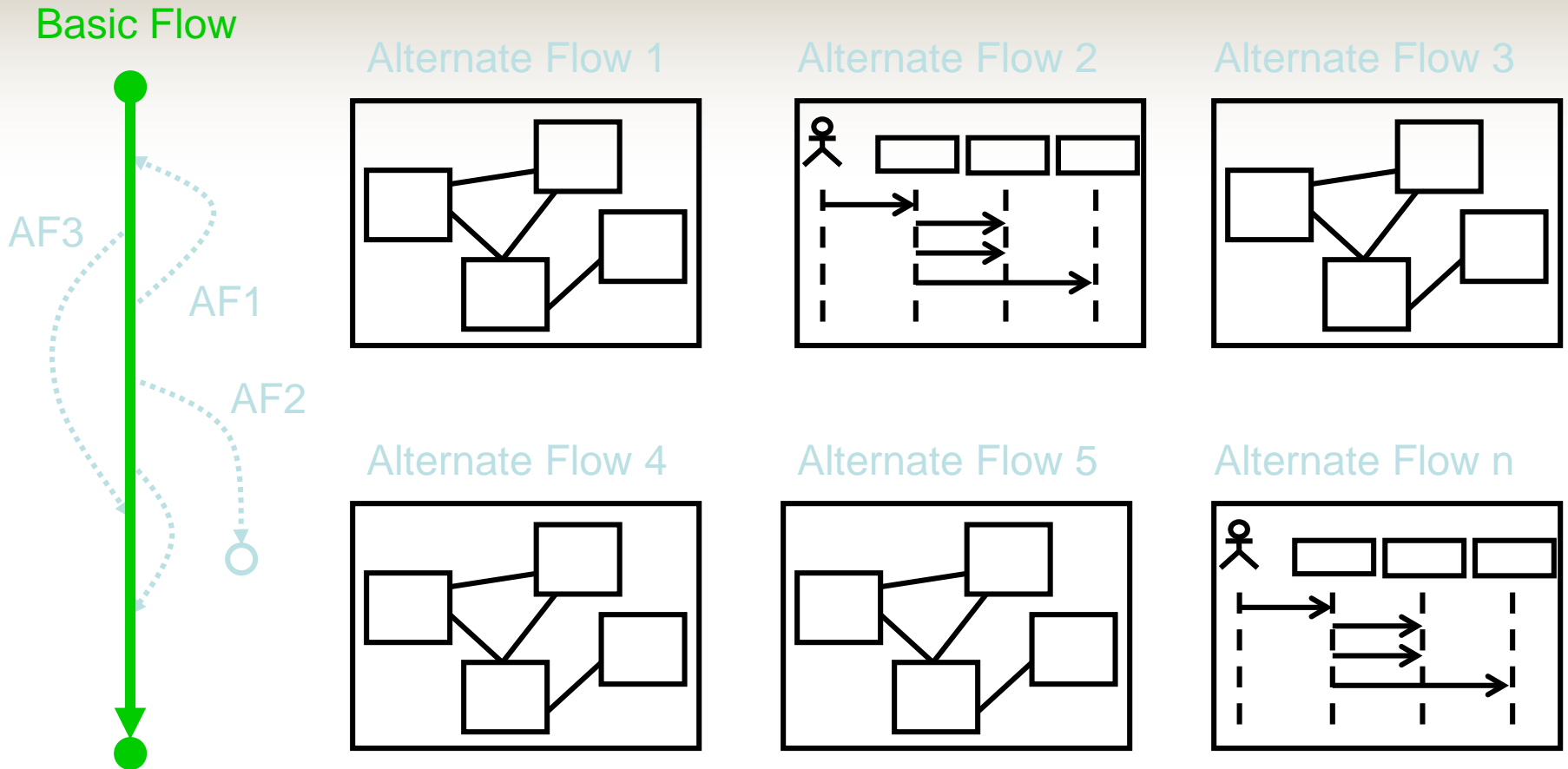
# Conditions and iterations - semantics

- ▶ The keyword "{new}" shown on an object indicates that the object is created during this scenario
- ▶ The keyword "{destroy}" shown on an object indicates that the object is destroyed during this scenario
- ▶ The keyword "{new}" shown on a message indicates that this message / stimulus is responsible for creating the object
- ▶ The keyword "{destroy}" shown on a message indicates that this message is responsible to destroy the object

# One Interaction Diagram Not Good Enough

- ▶ Model ‘**most** of the flows of events’ to make sure that all requirements needed are accommodated by the participating classes.
- ▶ → Start with describing the basic flow, which is the most common or most important flow of events.
- ▶ → Then describe variants such as exceptional flows.
  - ➔ You **do not have to describe** all the flows of events, as long as you employ and exemplify all **operations** of the participating objects; that is, include methods...
  - Trivial flows can be omitted, such as those that concern only one object.

# One Interaction Diagram Not Good Enough



**Note: Alternative paths return to main flow or else terminate. Must be explicit!**  
**Can all be shown visually with an Activity Diagram for that Use Case**

# Collaboration Diagrams Vs Sequence Diagrams

## Same information expressed in different ways...

### ► Collaboration Diagrams

- Show relationships in addition to interactions
- Better for visualizing patterns of collaboration
- Better for visualizing all of the effects on a given object
- Easier to use for brainstorming sessions

### ► Sequence Diagrams

- Show the explicit sequence of messages
- Better for visualizing overall flow
- Better for real-time specifications and for complex scenarios



# **Specifying Relationships**

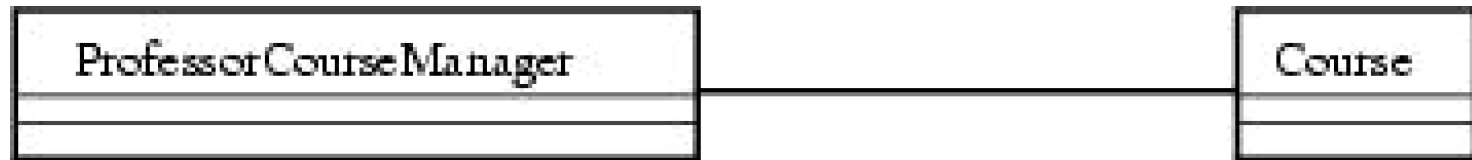
## **Chapter 6**

# Associations

- ▶ Two types of relationships discovered during analysis are associations and aggregations.
- ▶ An **association** is a bidirectional semantic connection between classes.
  - An association between classes means that there is a link between objects in the associated classes.
  - For example, an association between the Course class and the ProfessorCourseManager class means that objects in the Course class are connected to objects in the professorCourseManager class.
  - The number of objects connected depends upon the multiplicity of the association

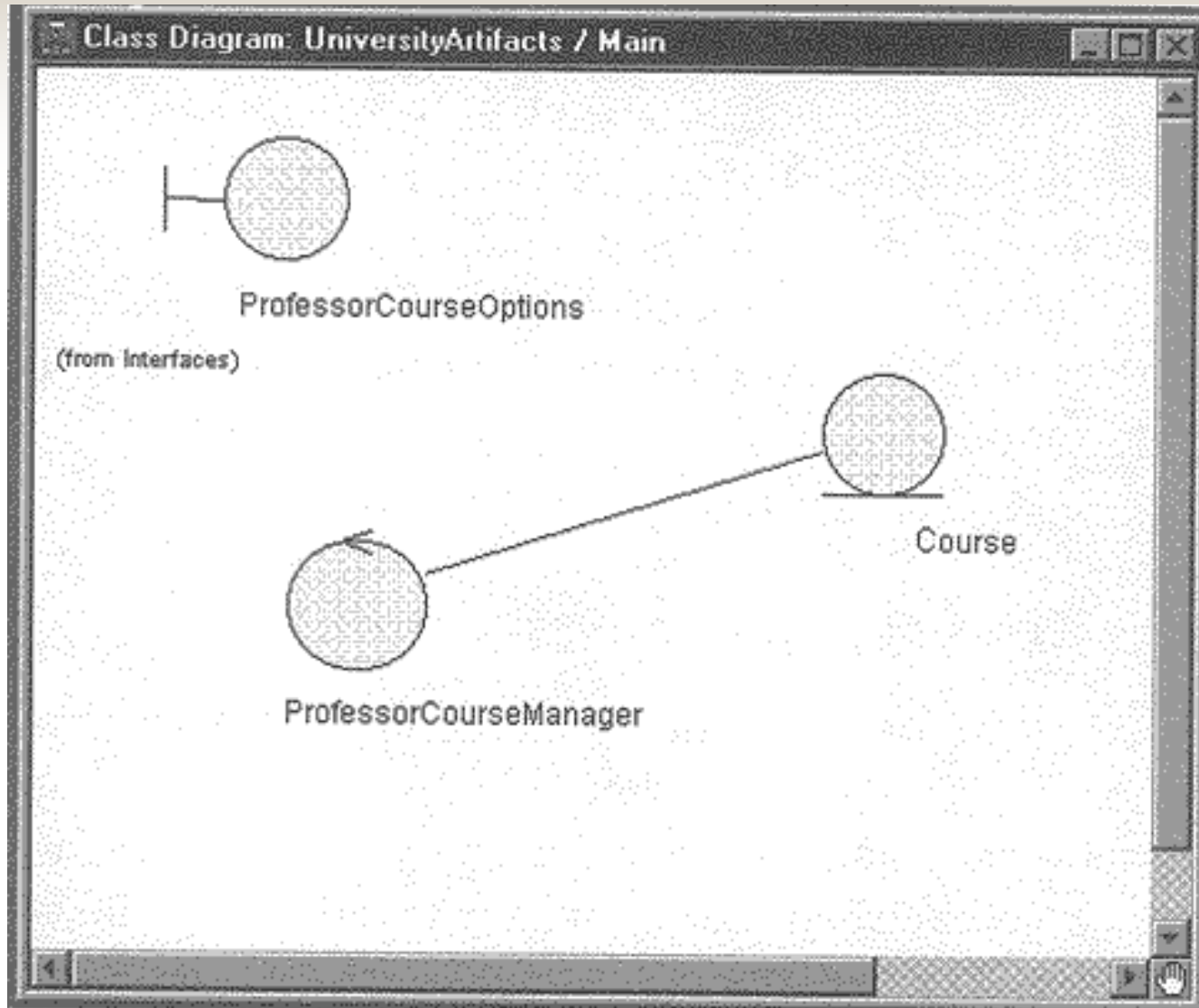
# Associations -2

- ▶ In the UML, association relationships are shown as a line connecting the associated classes, as shown



- ▶ **CREATING AN ASSOCIATION RELATIONSHIP IN RATIONAL ROSE**
  1. Click to select the Association icon from the toolbar. The association icon may be added to the toolbar by right-clicking on the toolbar and selecting the Customize menu command.
  2. Click on one of the associated classes in a class diagram.
  3. Drag the association line to the other associated class.

# Associations -3



# Aggregation Relationships

- ▶ An aggregation relationship is a specialized form of association in which a whole is related to its part(s).
  - Aggregation is known as a "part-of" or containment relationship.
  - The UML notation for an aggregation relationship is an association with a diamond next to the class denoting the aggregate (whole), as shown in Figure 6-3.



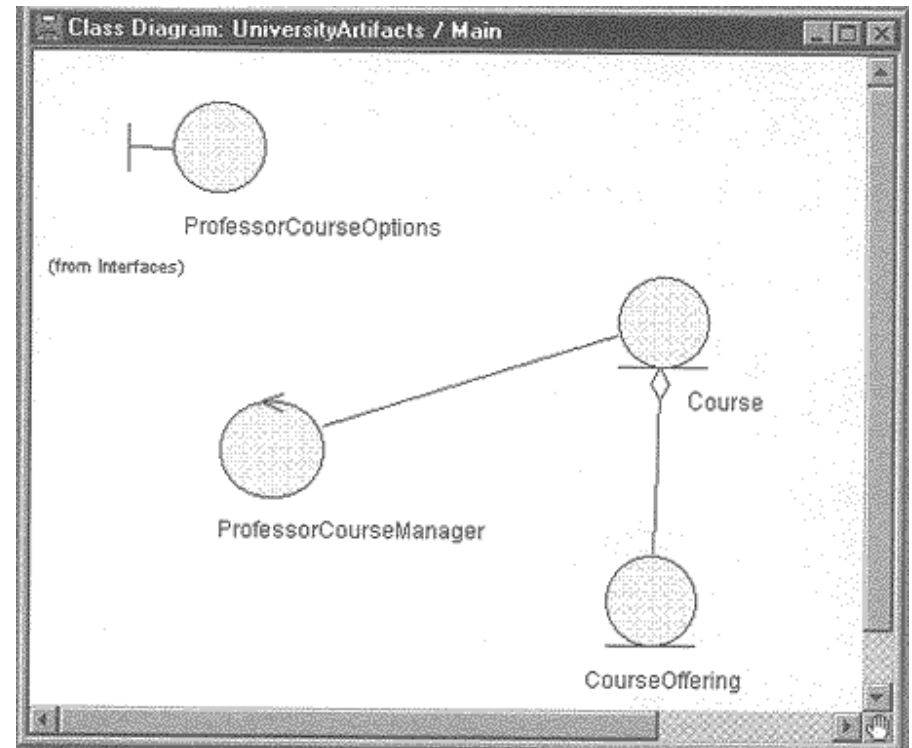
# Aggregation Relationships-2

- ▶ The following tests may be used to determine if an association should be an aggregation:
  - Is the phrase "part of" used to describe the relationship?
  - Are some operations on the whole automatically applied to its parts? For example, delete a course then delete all of its course offerings.
  - Is there an intrinsic asymmetry to the relationship where one class is subordinate to the other?
  - For example, a Course (Math 101) may be offered at different times during a semester. Each offering is represented as a Course Offering (e.g., Math 101, Section 1, and Math 101, section 2). The relationship between a Course and a CourseOffering is modeled as an aggregation—a Course "has" CourseOfferings.

# Aggregation Relationships-3

## ► CREATING AN AGGREGATION RELATIONSHIP IN RATIONAL ROSE

1. Select the Aggregation icon from the toolbar. The Aggregation icon may be added to the toolbar by right-clicking on the toolbar and selecting the Customize menu command.
2. Click on the class playing the role of the "whole" in a class diagram and drag the aggregation line to the class playing the role of the "part."



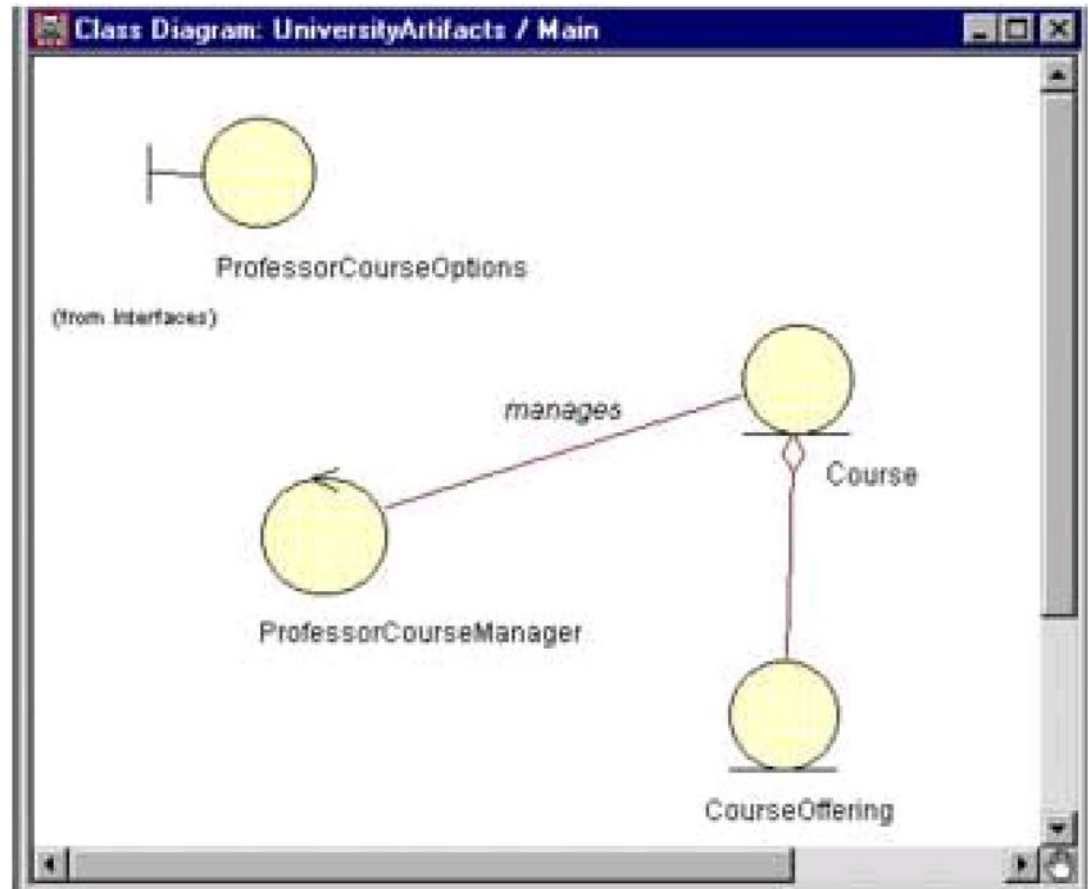
# Naming Relationships

- ▶ An association may be named.
- ▶ Usually the name is an active verb or verb phrase that communicates the meaning of the relationship.
- ▶ It is important to note that the name of the association is optional. Names are added if they are needed to add clarity to the model.
- ▶ Aggregation relationships typically are not named since they are read using the words "has" or "contains."



# Naming Relationships-2

- ▶ NAMING RELATIONSHIPS IN RATIONAL ROSE
  1. Click to select the relationship line on a class diagram.
  2. Enter the name of the relationship.
- ▶ A named relationship is shown in Figure 6-5.



# Role Names

- ▶ The end of an association where it connects to a class is called an association role.
  - Role names can be used instead of association names.
  - A role name is a noun that denotes the purpose or capacity wherein one class associates with another.
  - The role name is placed on the association near the class that it modifies, and may be placed on one or both ends of an association line.
  - It is not necessary to have both a role name and an association name.

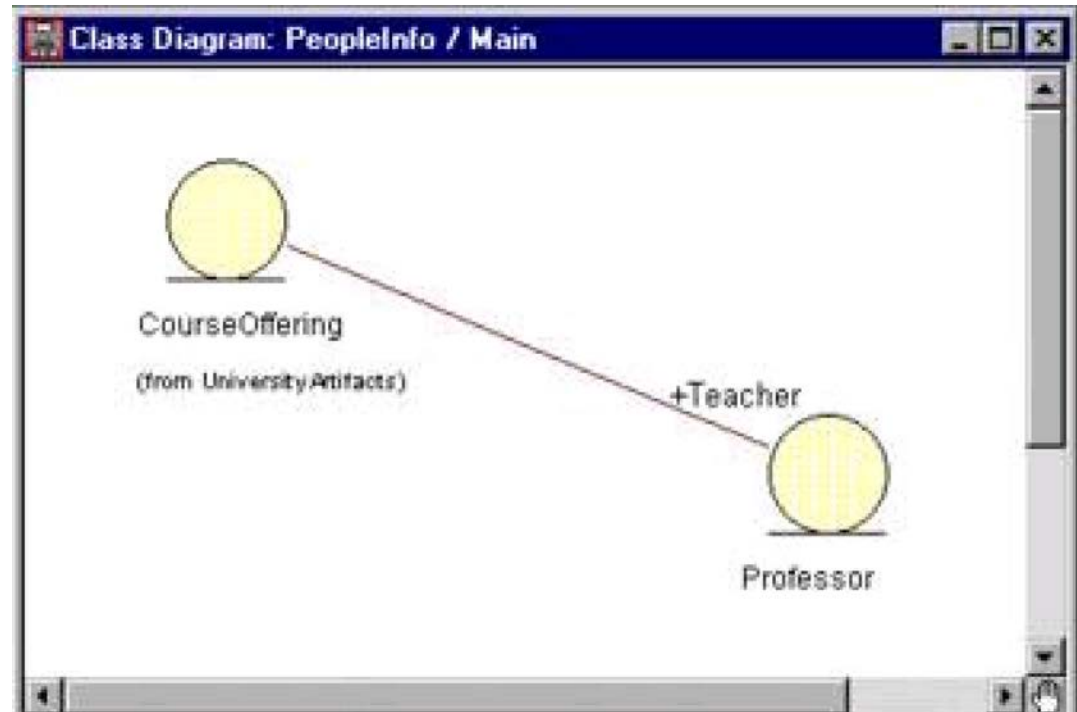
# Role Names-2

## ► CREATING ROLE NAMES IN RATIONAL ROSE

1. Right-click on the relationship line near the class that it modifies to make the shortcut menu visible.
2. Select the Role Name menu choice.
3. Enter the name of the role.

**The relationship shown in Figure 6-6 is read in both directions.**

- A Professor playing the role of the Teacher is related to the CourseOffering
- A CourseOffering is related to a Professor playing the role of a Teacher



# Multiplicity Indicators

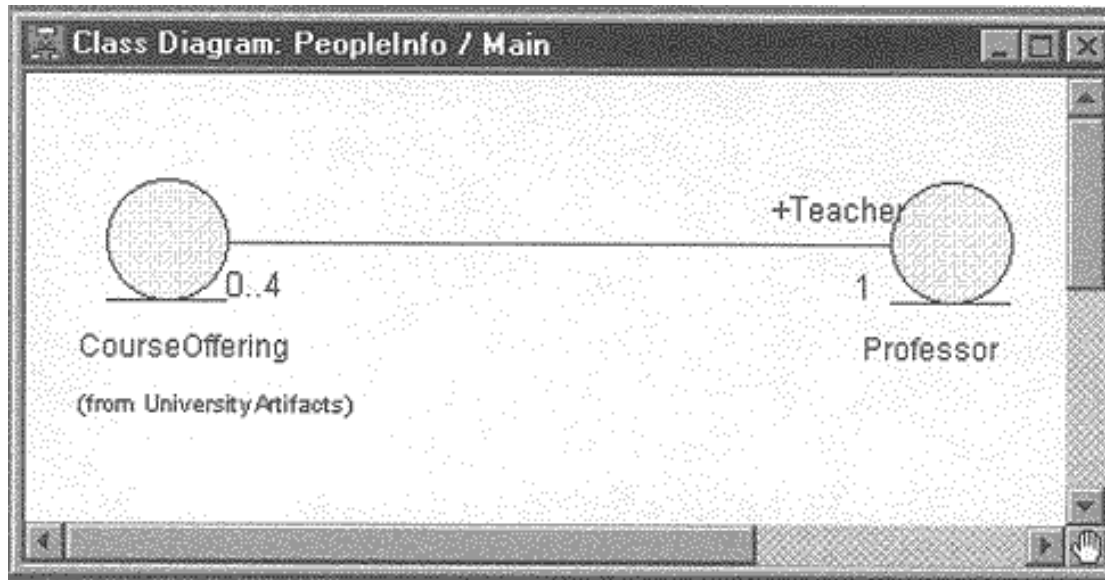
- ▶ Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship.
- ▶ Multiplicity defines the number of objects that are linked to one another.
- ▶ There are two multiplicity indicators for each association or aggregation—one at each end of the line.
- ▶ **Some common multiplicity indicators are**

1	Exactly one
0..*	Zero or more
1..*	One or more
0..1	Zero or one
5..8	Specific range (5, 6, 7, or 8)
4..7,9	Combination (4, 5, 6, 7, or 9)

# Multiplicity Indicators-2

## ► CREATING MULTIPLICITY IN RATIONAL ROSE

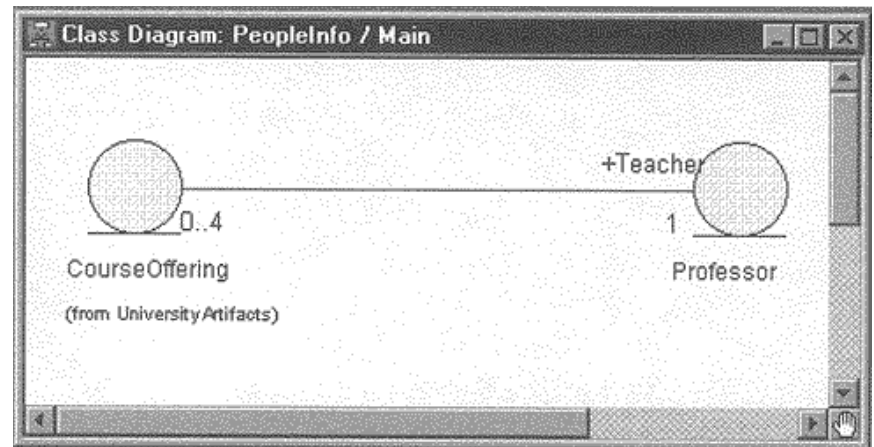
1. Double-click on the relationship line to make the Specification visible.
2. Select the Detail tab for the role being modified (Role A Detail or Role B Detail).
3. Enter the desired multiplicity in the Cardinality field.
4. Click the OK button to close the Specification.
5. Multiplicity indicators are shown in Figure 6-7.



# Multiplicity Indicators-2

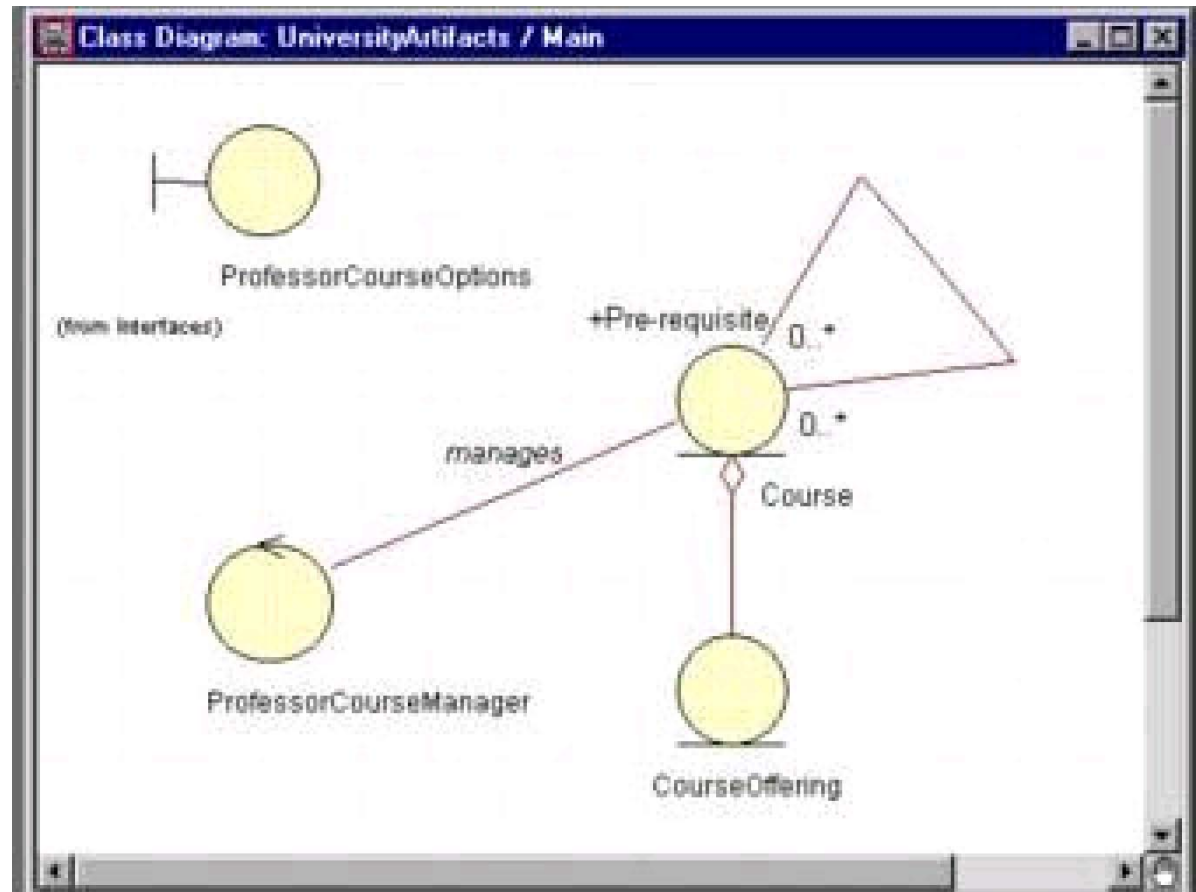
► The drawing in Figure 6-7 may be read in the following ways:

- One CourseOffering object is related to *exactly one* Professor object playing the role of the Teacher. For example, Math 101, Section 1 (a CourseOffering object) is related to Professor Smith (a Professor object).
- One Professor object playing the role of the Teacher is related to *zero to four* CourseOffering objects. For example, Professor Smith (a Professor object) is related to Math 101, Section 1; Algebra 200, Section 2; and Calculus 1, Section 3 (CourseOffering objects). Since the multiplicity is a range of zero to four, as few as zero CourseOffering objects to a maximum of four CourseOffering objects may be linked to one Professor object.



# Reflexive Relationships

- ▶ Multiple objects belonging to the same class may have to communicate with one another. This is shown on the class diagram as a reflexive association or aggregation.
- ▶ Role names rather than association names typically are used for reflexive relationships.



# Reflexive Relationships-2

## ► CREATING A REFLEXIVE RELATIONSHIP IN RATIONAL ROSE

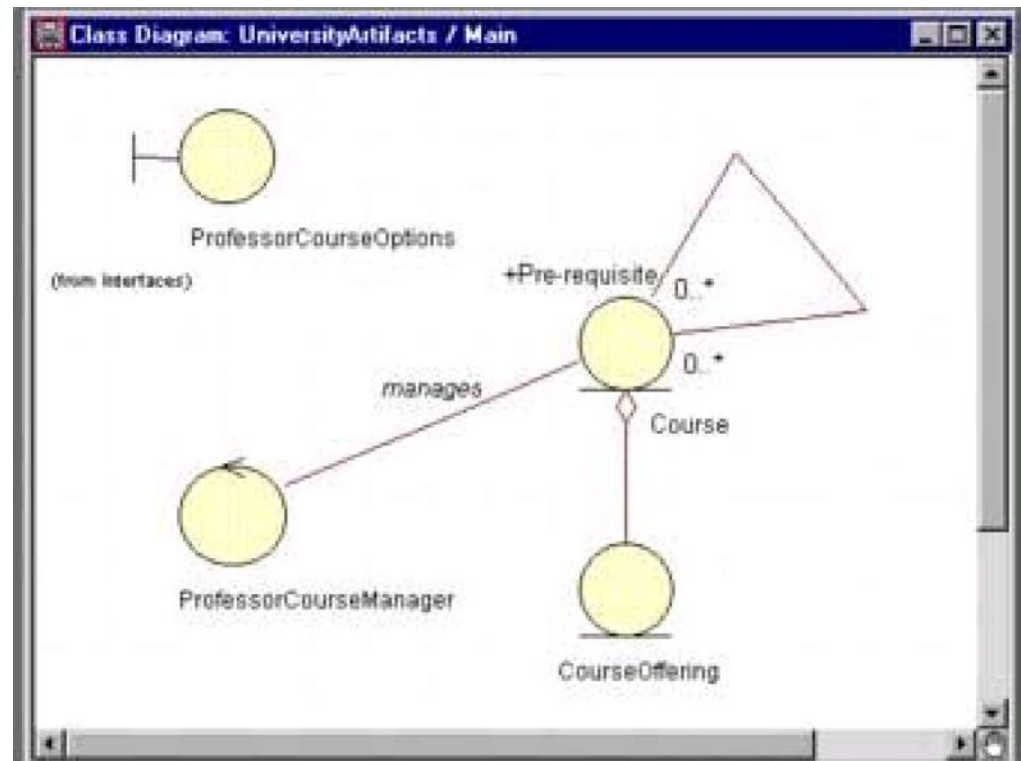
1. Select the Association (or Aggregation) icon from the toolbar.
2. Click on the class and drag the association (or aggregation) line outside the class.
3. Release the mouse button.
4. Click and drag the association (or aggregation) line back to the class.
5. Enter the role names and multiplicity for each end of the reflexive association (or aggregation).



# Reflexive Relationships-3

► The reflexive relationship in Figure 6-8 may be read in the following ways:

- One Course object playing the role of Prerequisite is related to zero or more Course objects.
- One Course object is related to zero or more Course objects playing the role of Prerequisite.



# Finding Relationships

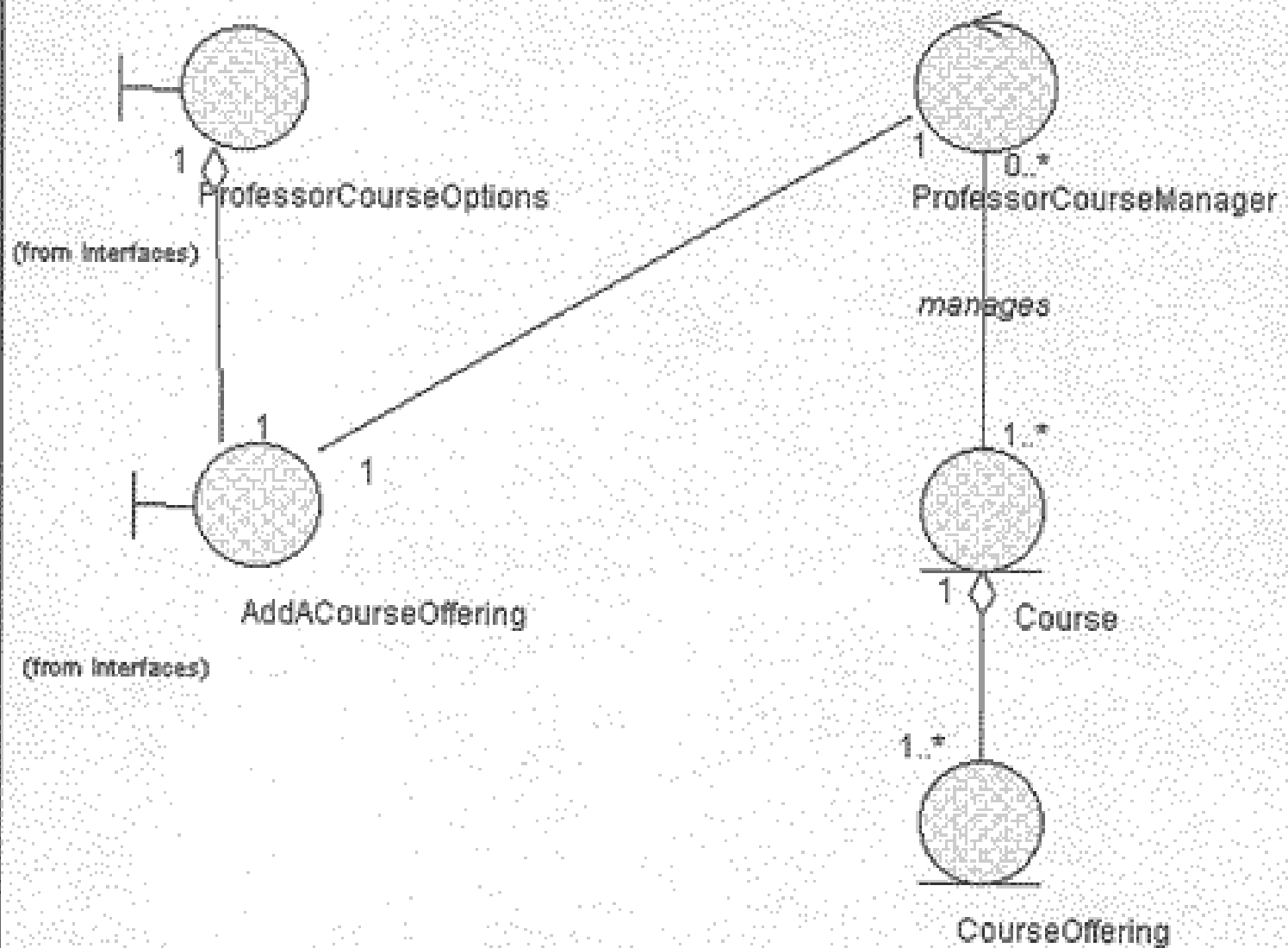
- ▶ Scenarios are examined to determine if a relationship should exist between two classes.
- ▶ Messages between objects mean that the objects must communicate with each other.
- ▶ Associations and/or aggregations provide the pathway for communication.
- ▶ Relationships may also be discovered based on the signature of an operation. This is discussed in Chapter 7.

# Relationships in the ESU Course Registration Problem

- Relationships in the ESU Course Registration Problem
- In the *Add a Course Offering* scenario, the communicating objects along with the relationship-type decisions that have been made are shown in Table 6-1.

Table 6-1. Class Relationships

Sending Class	Receiving Class	Relationship Type
ProfessorCourseOptions	AddACourseOffering	Aggregation
AddACourseOffering	ProfessorCourseManager	Association
ProfessorCourseManager	Course	Association
Course	CourseOffering	Aggregation



# Package Relationships

- ▶ Package relationships are also added to the model.
- ▶ The type of relationship is a dependency relationship, shown as a dashed arrow to the dependent package, as shown in Figure 6-10. If package A is dependent on package B this implies that one or more classes in package A initiates communication with one or more public classes in package B.
- ▶ Package A is referred to as the Client package and package B is referred to as the Supplier package.
- ▶ Package relationships are also discovered by examining the scenarios and class relationships for the system under development. As this is an iterative process, the relationships will change as analysis and design progresses.

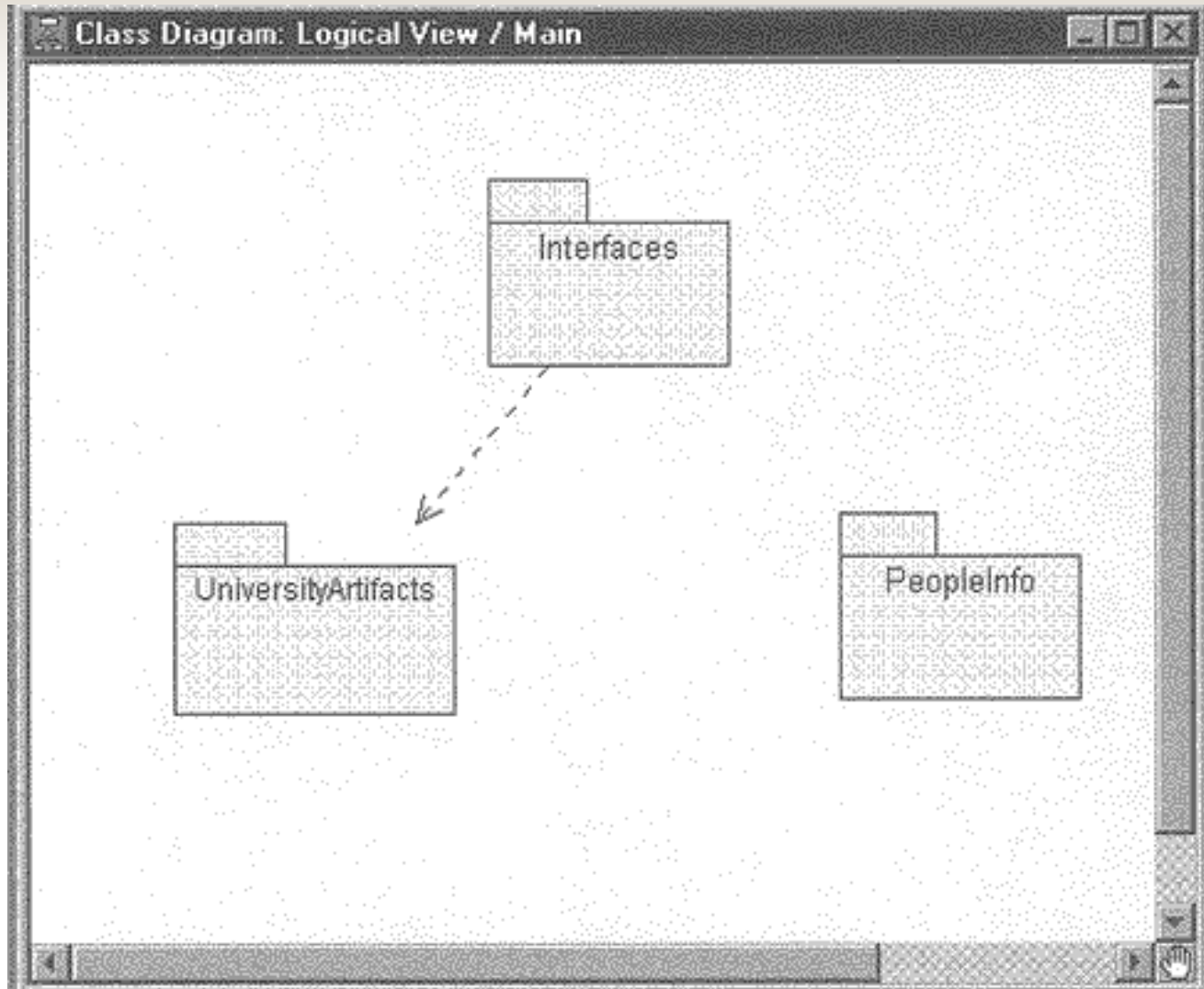
Figure 6-10. Package Relationships



# Package Relationships in the ESU Course Registration Problem

- ▶ In the *Add a Course Offering* scenario, the AddACourseOffering class sends a message to the
- ▶ ProfessorCourseManager class. This implies that there is a relationship between the Interfaces package and the UniversityArtifacts package. At this time, we have not discovered any relationships to the People package.
- ▶ **CREATING PACKAGE RELATIONSHIPS IN RATIONAL ROSE**
  1. Select the dependency relationship icon from the toolbar.
  2. Click on the client package and drag the arrow to the supplier package.

**The package relationships for the Course Registration System are shown in Figure 6-11.**





# Summary

- ▶ Relationships provide the conduit for object interaction.
- ▶ Two types of relationships between classes that are discovered during analysis are associations and aggregations.
- ▶ An association is a bidirectional semantic connection between classes.
- ▶ An aggregation is a specialized form of association in which a whole is related to its part(s).
- ▶ An association may be named. Usually the name is an active verb or verb phrase that communicates the meaning of the relationship.
- ▶ Roles can be used instead of association names. A role name is a noun that denotes the purpose or capacity wherein one class associates with another.
- ▶ Multiplicity is the number of instances that participate in a relationship. There are two multiplicity indicators for each association or aggregation—one at each end of the relationship line.
- ▶ Multiple objects belonging to the same class may have to communicate with one another. This is shown on the class diagram as a reflexive association or aggregation.
- ▶ Scenarios are examined to determine if a relationship should exist between two classes.
- ▶ Packages are related via dependency relationships. If package A is dependent on package B, this implies that one or more classes in package A initiates communication with one or more public classes in package B.

# Case Tools

**Chapter 7. Adding Behavior and Structure**

**Chapter 8. Discovering Inheritance**

**Chapter 9. Analyzing Object Behavior**

# Representing Behavior and Structure

- ▶ A class embodies a set of responsibilities that define the **behavior** of the objects in the class.
- ▶ The responsibilities are carried out by the operations defined for the class.
- ▶ The **structure** of an object is described by the attributes of the class.
- ▶ Each attribute is a data definition held by objects of the class.
- ▶ Objects defined for the class have a value for every attribute of the class.
  - For example, a Course class has the attributes of name, definition, and number of credit hours. This implies that every Course object will have a value for each attribute. Attribute values do not have to be unique—there are many three-credit courses in the university.

# Creating Operations

- Messages in interaction diagrams typically are mapped to operations of the receiving class.
- However, there are some special cases where the message does not become an operation.
  - If the receiving class is a boundary class that is a placeholder for a graphical user interface (GUI) type class, the message is a statement of the requirements for the GUI. These types of messages typically are implemented as some type of GUI control (i.e., a button) and are not mapped to operations since the behavior is carried out by the GUI control itself.
  - For example, the Professor actor must enter a password to start the *Add a Course Offering* scenario. This is represented as a message to the boundary class ProfessorCourseOptions. This will never be an operation of the user interface class—it will most likely be a text field on a window.
  - Messages to and from actors also receive special consideration. If the message is to or from an actor that represents a physical person, the message is a statement of human procedure and is therefore incorporated into a user manual, but not to an operation, since it does not make sense to create operations for humans. In the *Add a Course Offering* scenario, the fact that the Professor must have some sort of password to activate the system is an important requirement that should be captured in the user manual.
- If the message is to or from an actor that represents an external system, a class is created to hold the protocol that is used to perform the communication with the external system. In this case, the message is mapped to an operation of the class.

# Creating Operations-2

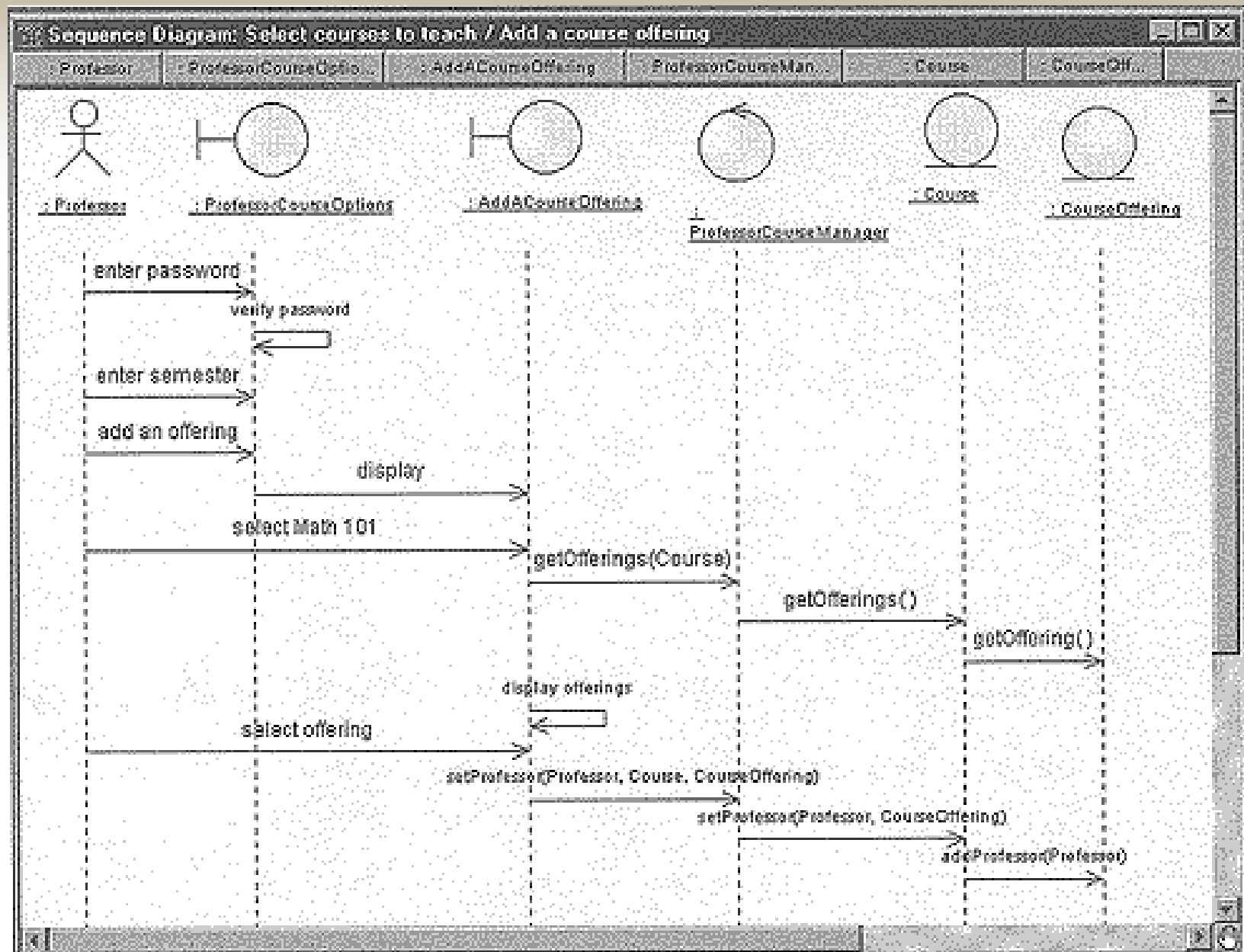
## ► MAPPING MESSAGES TO NEW OPERATIONS IN RATIONAL ROSE

1. Assign the objects to classes if that has not been done previously.
2. Right-click on the message arrow to make the shortcut menu visible.
3. Select the <new operation> menu choice. This will open the Operation Specification.
4. Enter the name of the operation in the Operation Specification.
5. Click the OK button to close the Operation Specification.

► Operations may also be created independently of interaction diagrams, since not all scenarios are represented in a diagram.

► This is also true for operations that are created to "help" another operation.

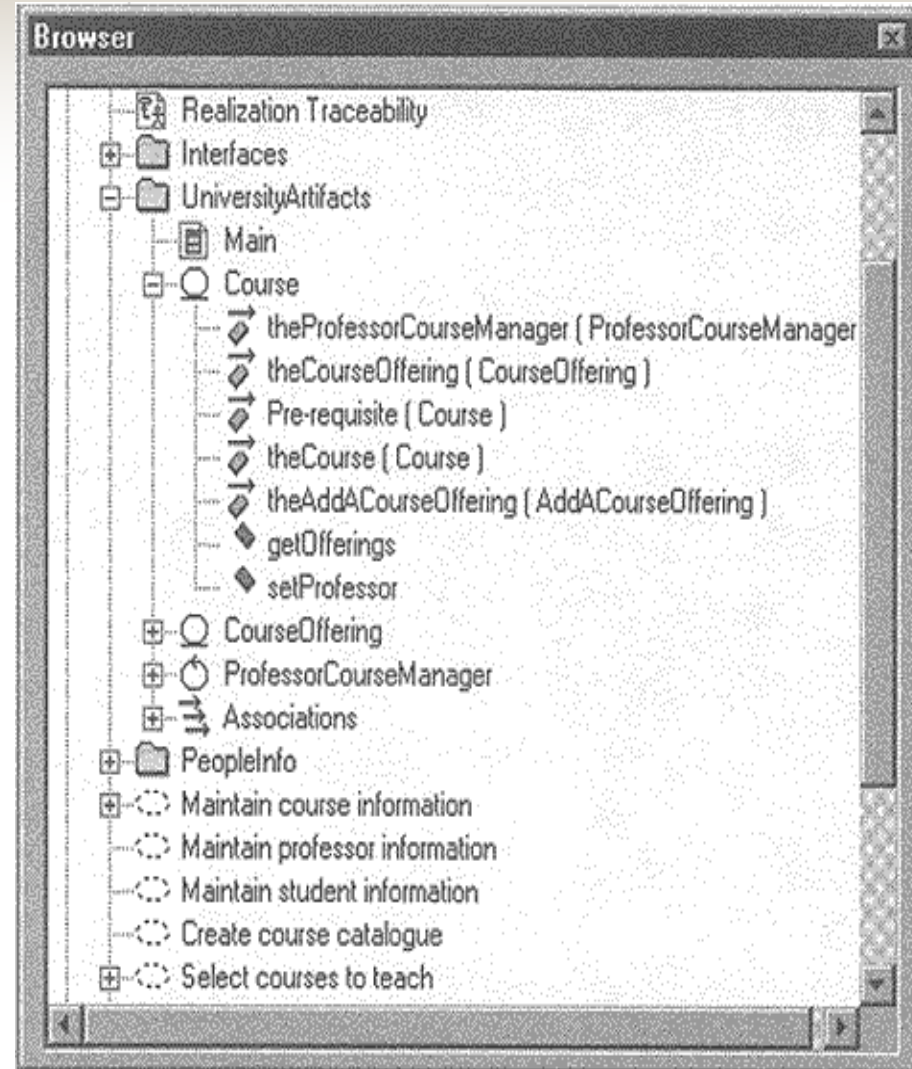
# Creating Operations-3



# Creating Operations-4

## ► CREATING OPERATIONS IN RATIONAL ROSE

1. Right-click to select the class in the browser and make the pop-up menu visible.
2. Select the New:Operation menu choice. This will create an operation called opname in the browser.
3. With the new operation selected, enter the desired name.
4. Operations for the Course class are shown in Figure 7-2.



# Relationships and Operation Signatures

- ▶ The signature of an operation may indicate a relationship.
- ▶ If the class for an argument of an operation or the return from an operation is a fundamental class such as a String, the relationship typically is not shown on a diagram.
- ▶ For other classes (i.e., nonfundamental classes) the relationship typically is displayed on one or more class diagrams.
  - For example, the two inputs to the setProfessor() operation of the Course class are professor (Professor class) and course offering (CourseOffering class). This implies that relationships exist between:
    - ▶ Course and Professor
    - ▶ Course and CourseOffering

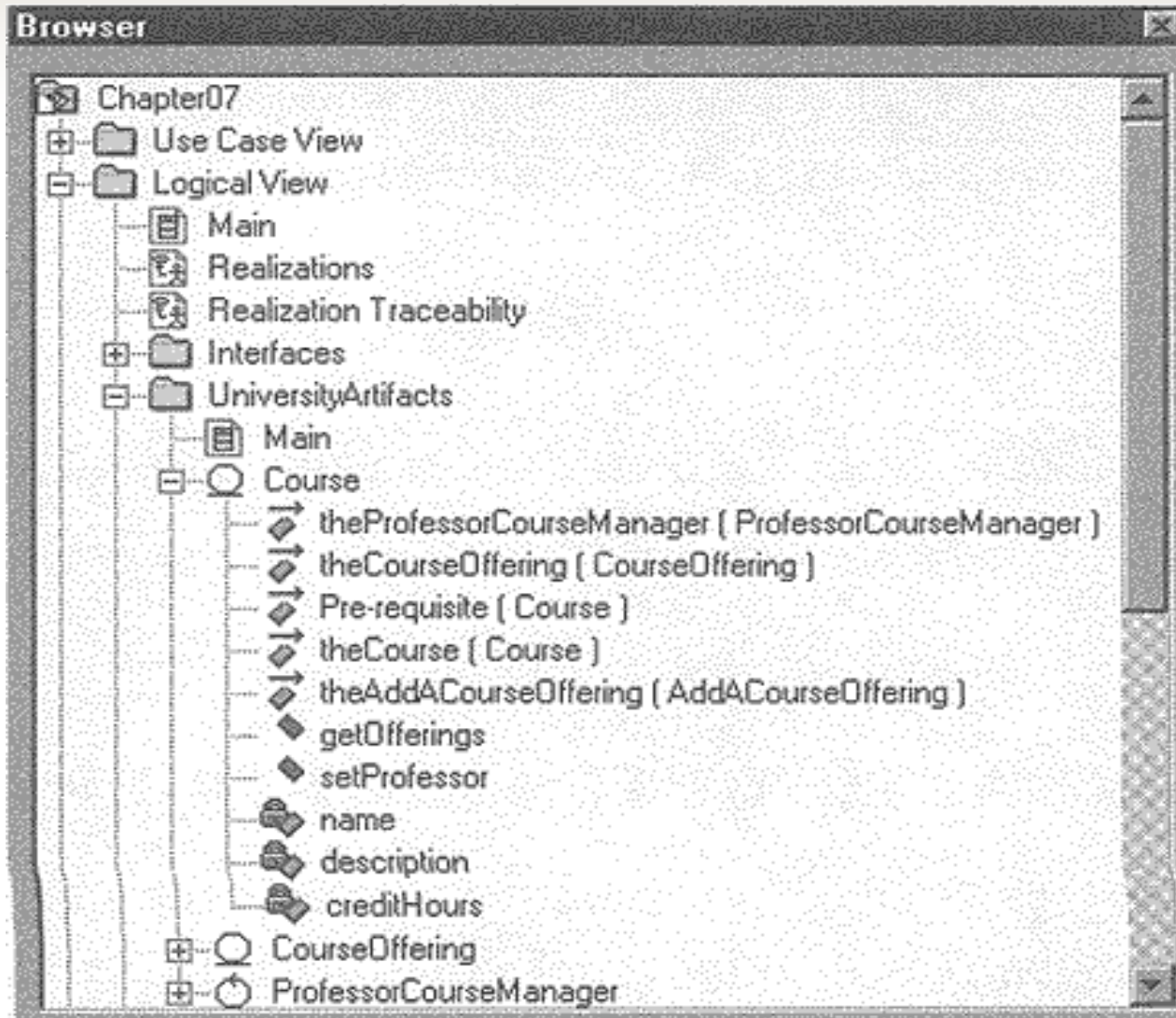


# Creating Attributes

- ▶ Many of the attributes of a class are found in the problem statement, the set of system requirements, and the flow of events documentation.
- ▶ They may also be discovered when supplying the definition of a class.
- ▶ Finally, domain expertise is also a good source of attributes for a class.
- ▶ For example, the requirements state that information such as course name, description, and number of credit hours is available in the Course Catalog for a semester. This implies that name, description, and number of credit hours are attributes of the Course class.
- ▶ **CREATING ATTRIBUTES IN RATIONAL ROSE**
  1. Right-click to select the class in the browser and make the pop-up menu visible.
  2. Select the New:Attribute menu choice. This will create an attribute called Name in the browser.
  3. With the new attribute selected, enter the desired name.

# Creating Attributes

- Attributes for the Course class are shown in Figure 7-4.



# Displaying Attributes and Operations

- ▶ Attributes and Operations may be displayed on a class diagram. Often, a class diagram is created specifically for this purpose—it shows the structure and behavior of the classes in a package. Relationships typically are not shown on this diagram.
- ▶ **CREATING A CLASS DIAGRAM TO SHOW THE ATTRIBUTES AND OPERATIONS FOR A PACKAGE**
  1. Right-click to select the package in the browser and make the shortcut menu visible.
  2. Select the New:Class Diagram menu choice. A class diagram called NewDiagram will be added to the browser.
  3. With the new diagram selected, enter the name of the diagram.
- ▶ **ADDING CLASSES TO A DIAGRAM USING THE QUERY MENU**
  1. Double-click on the diagram in the browser to open the diagram.
  2. Select the Query:Add Classes menu choice.
  3. Select the desired package. Click to select the desired classes and click the >>>> button to add the classes to the diagram or click the All >> button to add all the classes to the diagram.

# Displaying Attributes and Operations-1

## ► FILTERING RELATIONSHIPS IN RATIONAL ROSE

1. Double-click on the diagram in the browser to open the diagram.
2. Select the Query:Filter Relationships menu choice.
3. Click the None button in the Type field to hide all relationships shown on the open diagram.
4. Click the OK button to close the Relations window.

## ► DISPLAYING SOME ATTRIBUTES OR OPERATIONS IN RATIONAL ROSE

1. Right-click to select the class on an open class diagram and make the shortcut menu visible.
2. Select the Options:Select Compartment Items menu choice.
3. Click to select the attributes and operations to be displayed.
4. Click the >>>> button.
5. Click the OK button to close the Edit Compartment window.

# Displaying Attributes and Operations-2

## ► SHOWING ALL ATTRIBUTES AND OPERATIONS IN RATIONAL ROSE

1. Right-click on the class in a diagram to make the shortcut menu visible.
2. Select the Options:Show All Attributes menu choice to display all the attributes for the class.
3. Repeat step 1 and select the Options:Show All Operations menu choice to display all the operations for the class.

### ■ Note

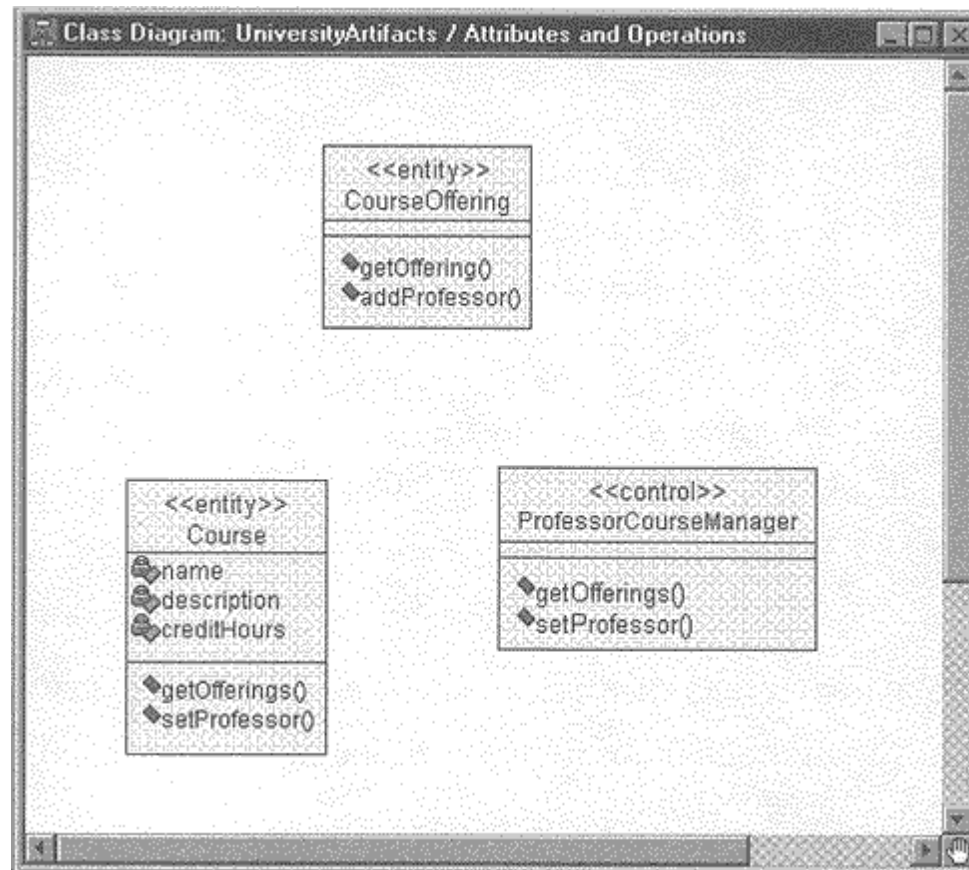
- To always display the attributes and operations for a class, you can set the Show All Attributes and Show All Operations selections using the Tools:Options menu.

## ► SETTING STEREOTYPE DISPLAY IN RATIONAL ROSE

1. Right-click on the class in a diagram to make the shortcut menu visible.
2. Select the desired Options:Stereotype Display menu choice (None = do not display stereotype, Label = show stereotype in << >>, Icon = show class using Stereotype icon).

# Displaying Attributes and Operations-3

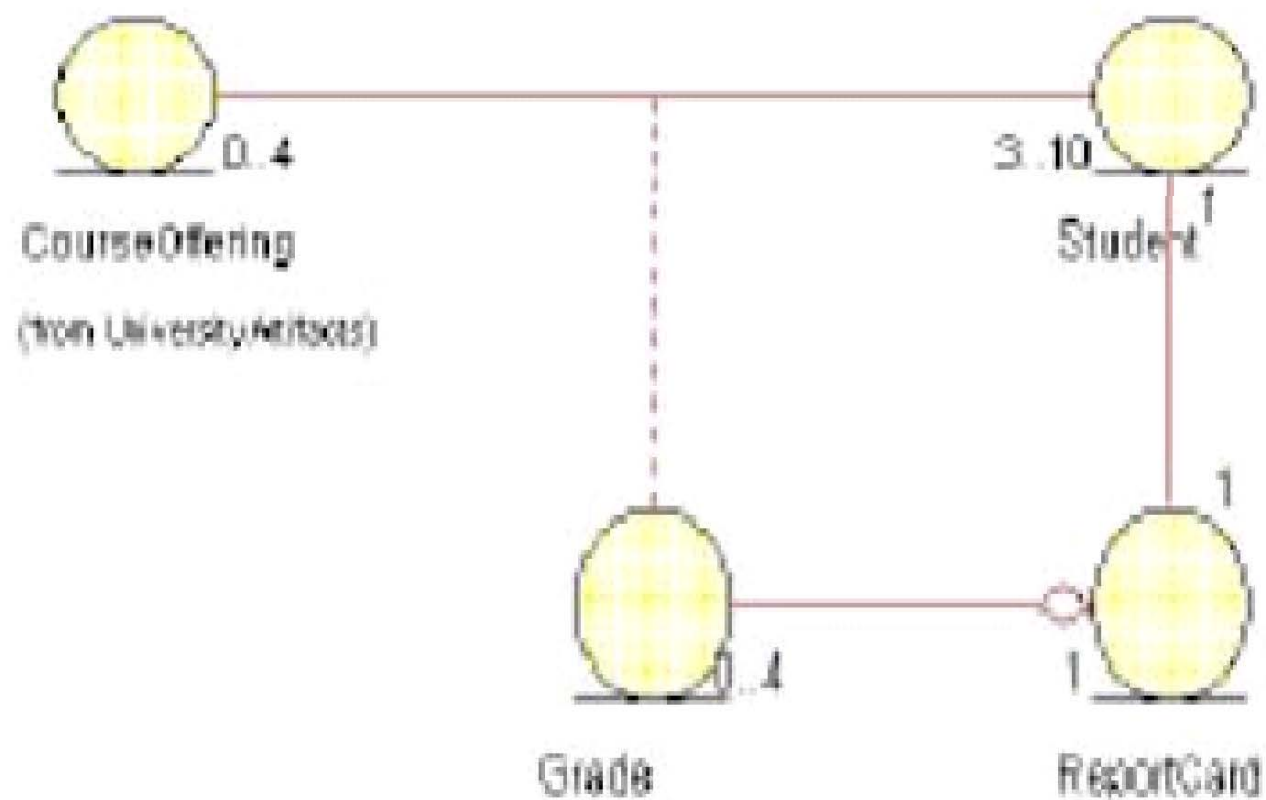
- The class diagram called Attributes and Operations for the University Artifacts package is shown in Figure 7-6. For this type of diagram I prefer to show the stereotypes of the classes as labels.



# Association Classes

- ▶ A relationship may also have structure and behavior. This is true when the information deals with a link between two objects and not with one object by itself.
- ▶ Consider the following example: A student may take up to four course offerings, and course offering may have between three and ten students. Each student must receive a grade for the course offering. Where is the grade held? It doesn't belong to the student since a student will probably have different grades for different course offerings, and it doesn't belong to the course offering since a course offering has different grades for different students.
- ▶ The information belongs to the link between a student and a course offering.
- ▶ This is modeled as an association class. Since an association class behaves like any other class, it too can have relationships.
- ▶ Following our example, a student receives a report card each semester that is made up of all the linked grade objects for the student.

# Class Diagram: PeopleInfo / Student Grades





# Summary

- ▶ A class embodies a set of responsibilities that define the behavior of the objects in the class. The responsibilities are carried out by the operations defined for the class.
- ▶ The structure of an object is described by the attributes of the class. Each attribute is a data definition held by objects of the class. Objects defined for the class have a value for every attribute of the class.
- ▶ The attributes and operations defined for a class are the ones that have meaning and utility within the application that is being developed.
- ▶ Messages in interaction diagrams typically are mapped to operations of the receiving class. However, there are some special cases where the message does not become an operation: messages to and from actors representing people and messages to and from classes representing GUI classes.
- ▶ Many of the attributes of a class are found in the problem statement, the set of system requirements, and the flow of events documentation. They may also be discovered when supplying the definition of a class. Finally, domain expertise is also a good source of attributes for a class.
- ▶ A relationship may also have structure and behavior. This is true when the information deals with a link between two objects and not with one object by itself.
- ▶ The structure and behavior belonging to a relationships is held in an association class.

# **Chapter 8.**

## **Discovering Inheritance**



# Inheritance

- ▶ Inheritance defines a relationship among classes where one class shares the structure and/or behavior of one or more classes.
- ▶ A hierarchy of abstractions is created in which a subclass inherits from one or more superclasses.
- ▶ Inheritance is also called an "is-a" or "kind-of" hierarchy.
- ▶ A subclass will inherit all attributes, operations, and relationships defined in any of its superclasses.
- ▶ Thus, attributes and operations are defined at the highest level in the hierarchy at which they are applicable, which allows all lower classes in the hierarchy to inherit them.
- ▶ Subclasses may be augmented with additional attributes and operations that apply only to that level of the hierarchy.
- ▶ A subclass may supply its own implementation of an operation.
- ▶ Since an inheritance relationship is not a relationship between different objects, the relationship is never named, role names are not used, and multiplicity does not apply.
- ▶ There is no limit to the number of classes allowed in an inheritance hierarchy. I
- ▶ inheritance is the key to reuse. A class can be created for one application and then a subclass may be created to add more information needed for a different application.



# Finding Inheritance

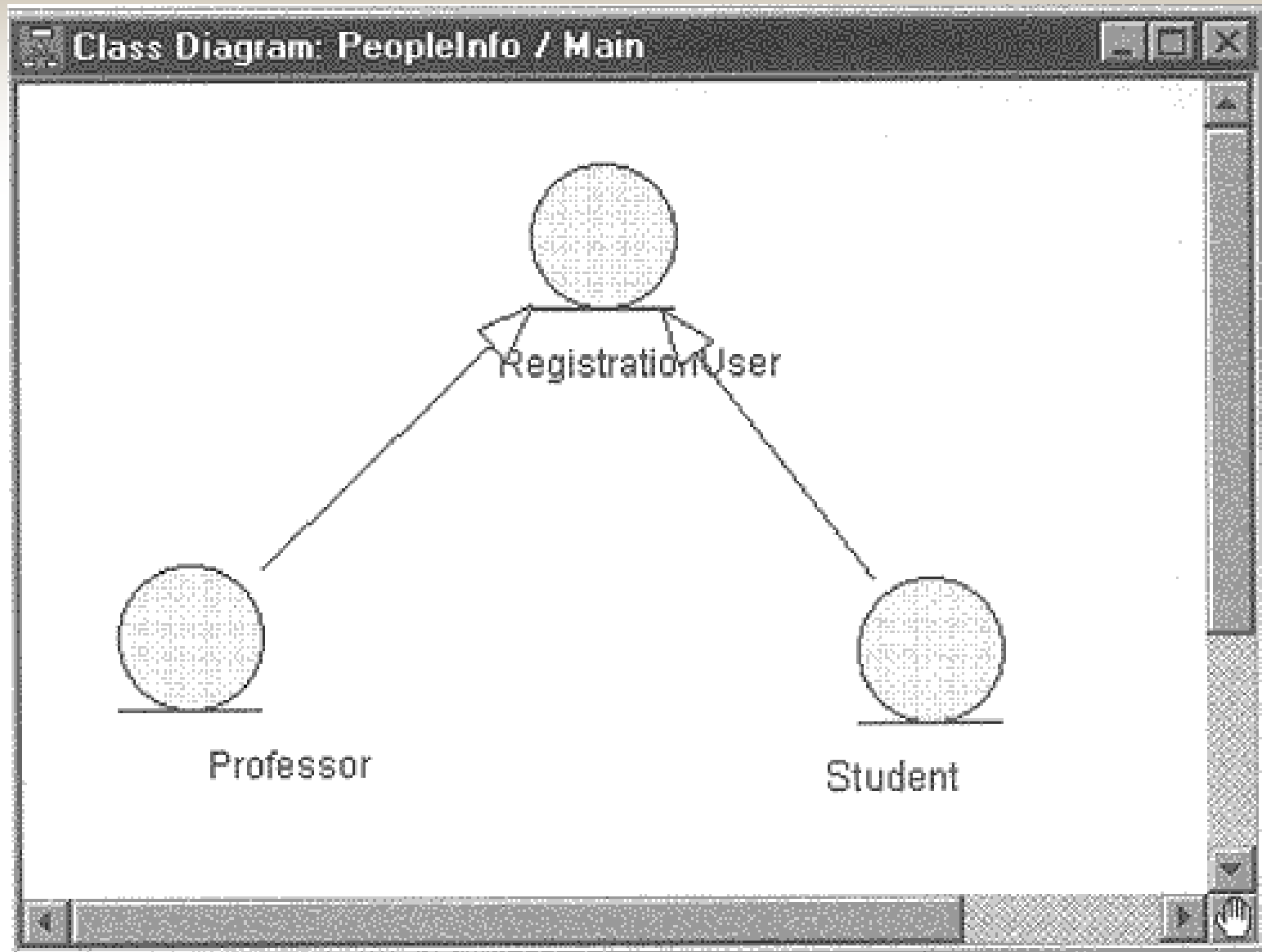
- ▶ There are two ways to find inheritance—generalization and specialization.
- ▶ Both methods typically are used for any system under development
- ▶ **Generalization** provides the capability to create superclasses that encapsulate structure and behavior common to several classes.
  - Classes are examined for commonality of structure (attributes) and behavior (operations). For example, the Student and Professor classes both have name, address, and phoneNumber as attributes.
- ▶ **Specialization** provides the ability to create subclasses that represent refinements to the superclass—typically, structure and behavior are added to the new subclass.
  - This method of finding inheritance often comes into play if a class already exists. Subclasses are added to specialize the behavior of an existing class. Operations may be over ridden by a subclass.
  - The subclass should never provide less behavior or structure than its superclasses

# CREATING INHERITANCE IN RATIONAL ROSE

## ► CREATING INHERITANCE IN RATIONAL ROSE

1. Open the class diagram that will display the inheritance hierarchy.
2. Click to select the Class icon from the toolbar and click on the open class diagram to draw the class.
3. With the class still selected, enter the name of the class.  
**Note: The class could also be created in the browser and added to the open class diagram.**
4. Click to select the Generalization icon on the toolbar.
5. Click on a subclass and drag the generalization line to the superclass.
6. Repeat step 5 for each additional subclass.

An inheritance relationship is shown in Figure 8-1.



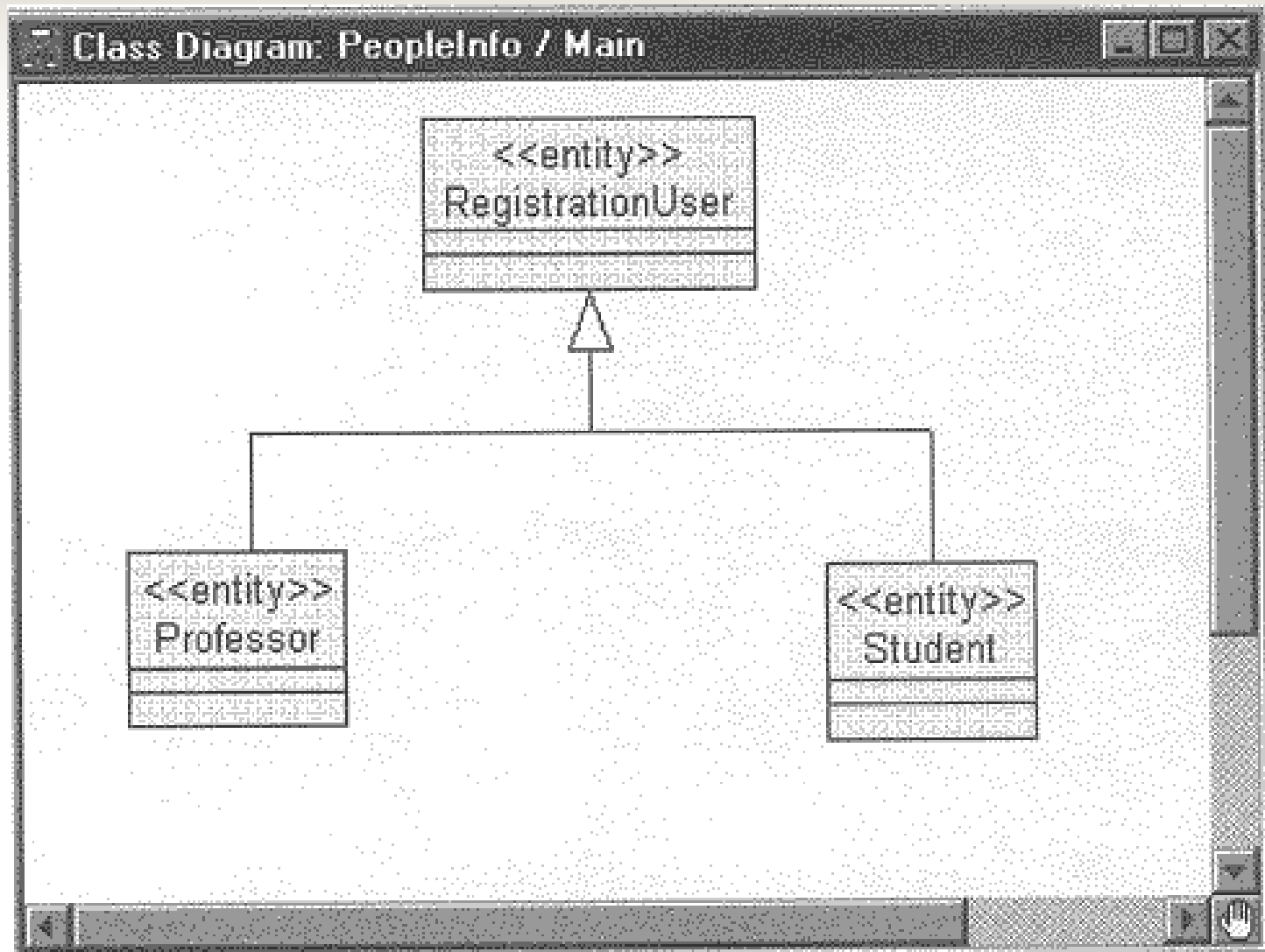
# Inheritance Trees

## ► CREATING AN INHERITANCE TREE IN RATIONAL ROSE

1. Open the class diagram that will display the inheritance hierarchy.
2. Click to select the Class icon from the toolbar and click on the open class diagram to draw the class.
3. With the class still selected, enter the name of the class.
  1. **Note** :The class could also be created in the browser and added to the open class diagram.
4. Click to select the Generalization icon on the toolbar.
5. Click on one subclass and drag the generalization line to the superclass.
6. For each subclass that is part of the inheritance tree, select the Generalization icon from the toolbar, click on the subclass, and drag the generalization line to the inheritance triangle.

- **Note** : An inheritance tree may be created from two separate generalization arrows by selecting one arrow and dragging it onto the other arrow.

An inheritance tree relationship is shown in Figure 8-2.



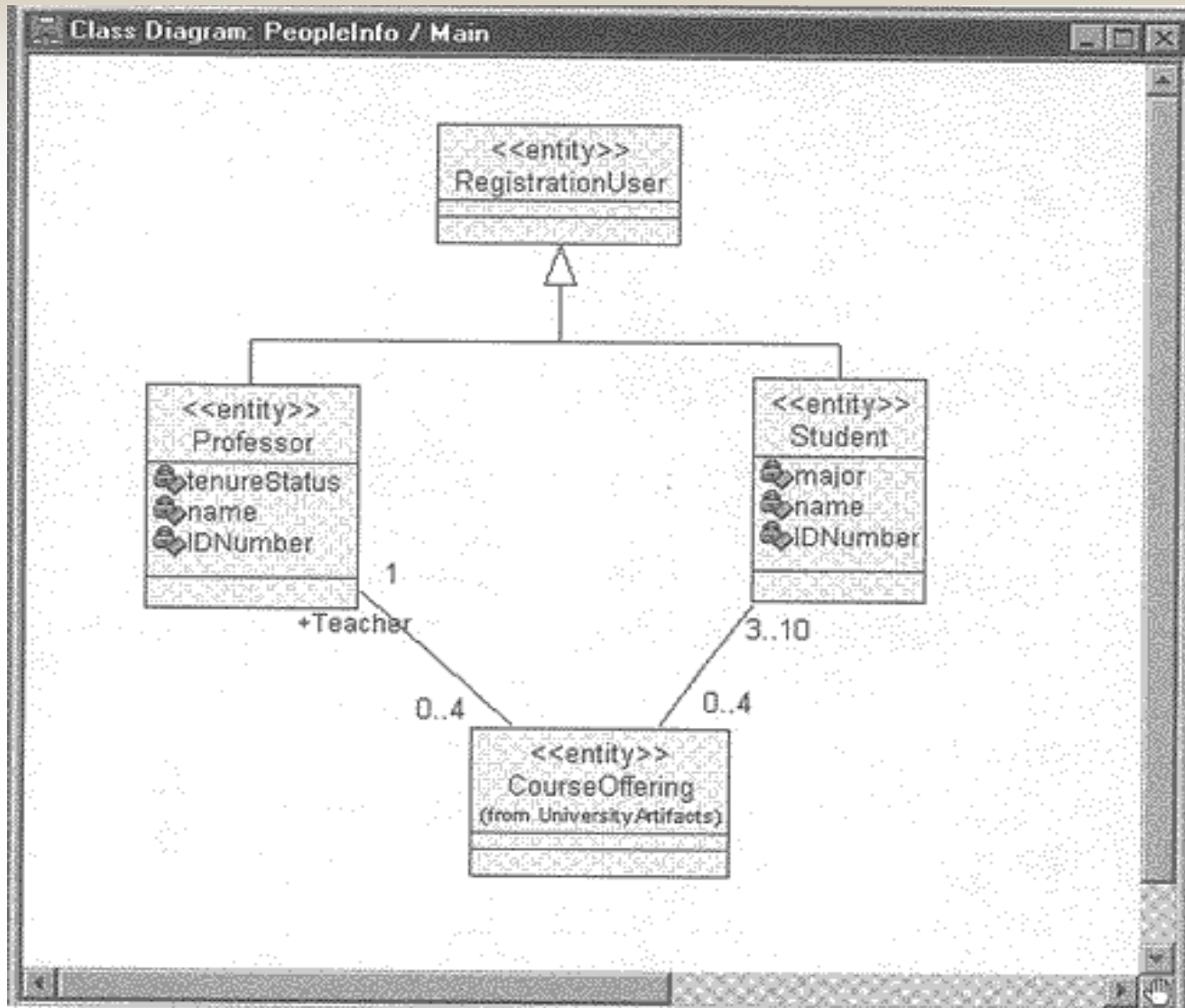




## Figure 8-3. RegistrationUser Inheritance Hierarchy

- ▶ Attributes, operations, and relationships are relocated to the highest applicable level in the hierarchy once a superclass is created.
- ▶ What features should be relocated? Let's look at the RegistrationUser hierarchy. The attributes, operations, and relationships for the subclasses are shown in Figure 8-3. As long as name and IDNumber have the same format they may safely be moved to the superclass (RegistrationUser).
- ▶ Both classes have a relationship to the CourseOffering class. There are two options that may be chosen for this relationship:
  - Keep the relationships at the subclass level.
  - Have one relationship at the superclass level with a multiplicity that includes the professor and the student objects (i.e., one CourseOffering object would be related to 4–11 RegistrationUser objects). Here a constraint stating that one RegistrationUser object must be a Professor object should be added to the model.

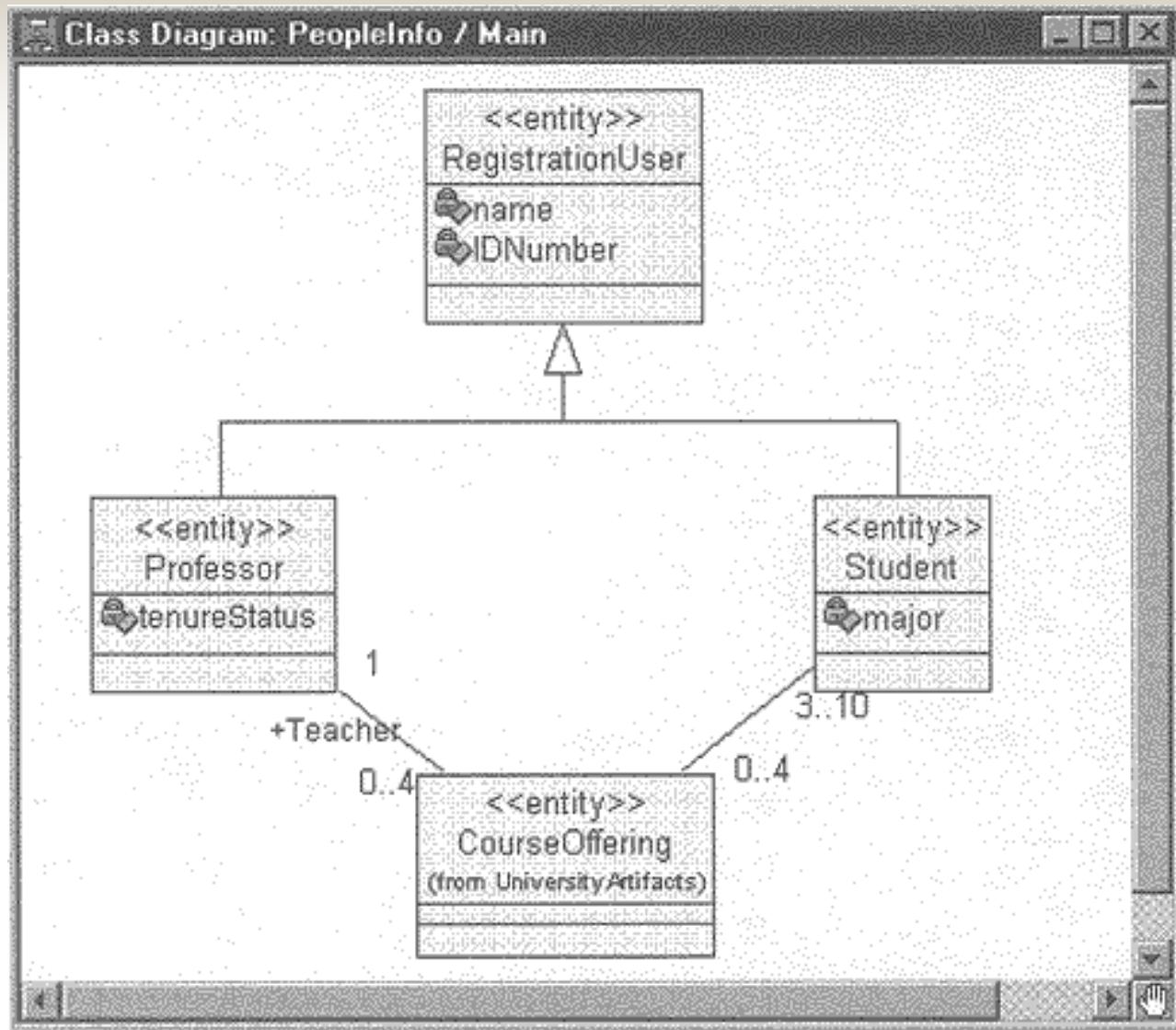
# Figure 8-3. Registration User Inheritance Hierarchy



## ► RELOCATING ATTRIBUTES AND OPERATIONS IN RATIONAL ROSE

1. Click the + sign next to one subclass in the browser to expand the class.
2. Select the attribute or operation to be relocated.
3. Drag the attribute or operation to the superclass.
4. Delete the attribute or operation from all other subclasses.
5. Repeat steps 2 through 4 for each additional attribute or operation to be relocated.

Relocated attributes are shown in Figure 8-4.



# Chapter 9.

## Analyzing Object Behavior

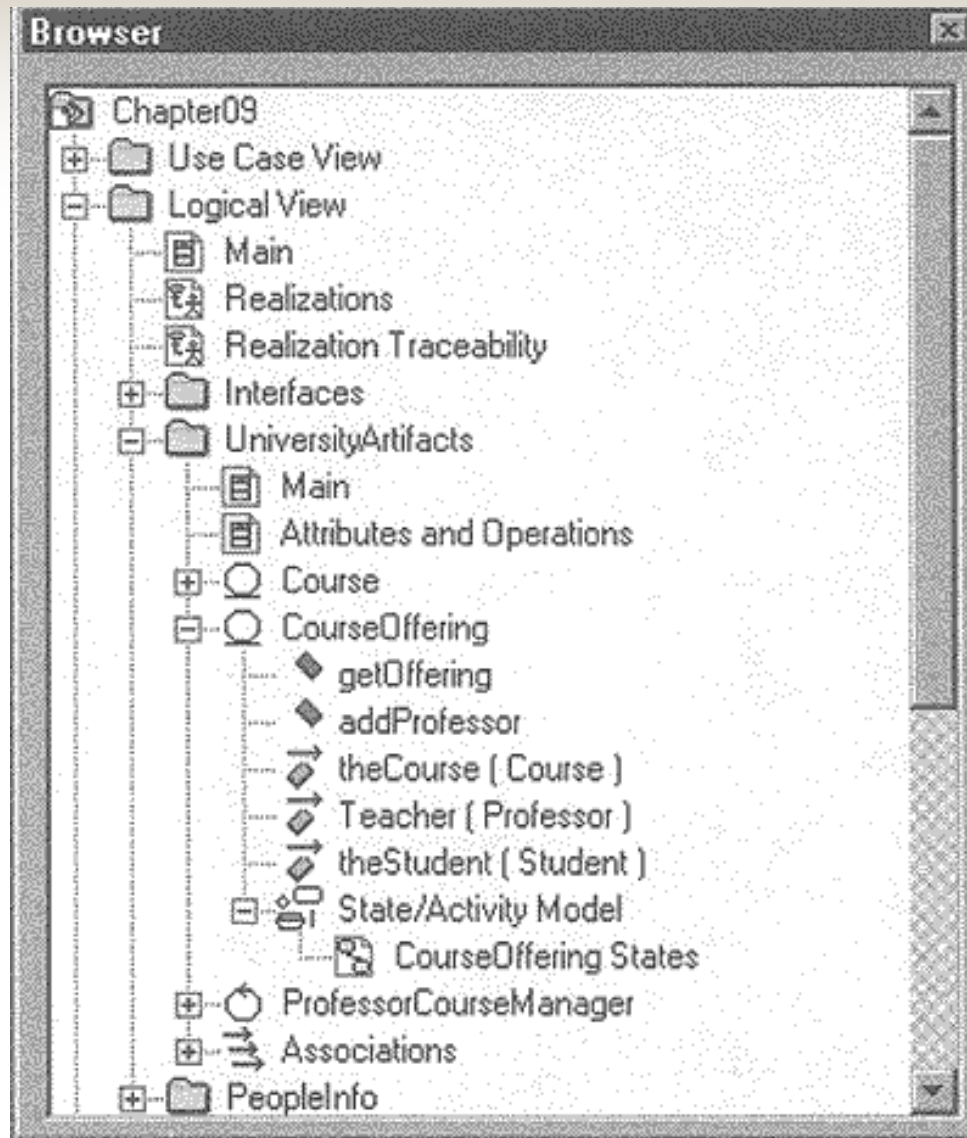
# Modeling Dynamic Behavior

- ▶ Use cases and scenarios provide a way to describe system behavior; that is, the interaction between objects in the system.
- ▶ Sometimes it is necessary to look at the behavior inside an object.
- ▶ A statechart diagram shows the **states** of a single object, the events or **messages** that cause a transition from one state to another, and the **actions** that result from a state change.
- ▶ A statechart diagram will not be created for every class in the system, only for classes with "significant" dynamic behavior.
- ▶ Interaction diagrams can be studied to determine the dynamic objects in the system—ones receiving and sending many messages.
- ▶ Statechart diagrams are also useful to investigate the behavior of an aggregate "whole" class and of control classes.
- ▶ Care must be taken to stay in an analysis frame of mind—concentrating on the WHAT of the problem and not the HOW of the solution.

# CREATING STATECHART DIAGRAMS IN RATIONAL ROSE

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the New:Statechart Diagram menu choice. This will add a state diagram called NewDiagram to the browser.
3. While the diagram is still selected, enter the name of the diagram.
4. To open the diagram, click the + to expand the class in the browser, click the + to expand the State/Activity Model in the browser and double-click on the statechartdiagram in the browser.

The browser view of the statechart diagram for the CourseOffering class is shown in Figure 9-1.





# States

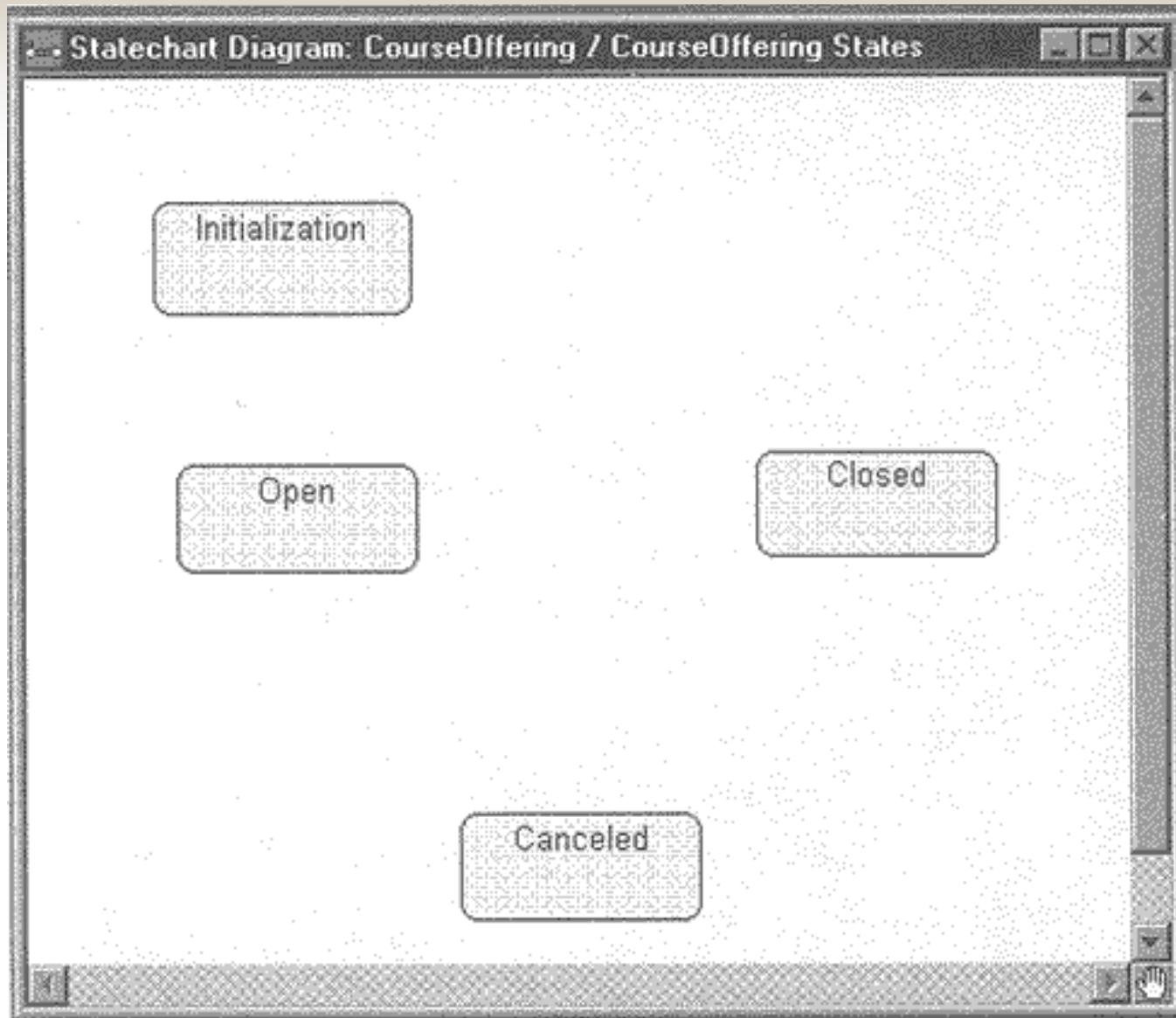
- ▶ A state is a condition during the life of an object during which it satisfies some condition, performs some action, or waits for an event.
- ▶ The state of an object may be characterized by the value of one or more of the attributes of the class.
  - For example, a CourseOffering object may be open (able to add a student) or closed (maximum number of students already assigned to the CourseOffering object).
  - The state depends upon the number of students assigned to the particular CourseOffering object.
- ▶ Additionally, a state of an object may be characterized by the existence of a link to another object.
  - A professor may be teaching or on sabbatical. This depends upon the existence of a link to a CourseOffering object.
  - Looking at the state of an object can validate the multiplicity chosen for a relationship to another object. That is, if being in a state depends upon the existence of a link to another object, this implies that the multiplicity of the relationship modifying the role of the associated class must include zero (i.e., the relationship is optional). Thus, the states of an object are found by examining the attributes and links defined for the object.
- ▶ The UML notation for a state is a rectangle with rounded corners as shown in Figure 9-2.



# Statechart diagram

- ▶ A statechart diagram encompasses all the messages that an object can send and receive.
- ▶ Scenarios represent one path through a statechart diagram.
- ▶ The interval between two messages sent by an object typically represents a state. Therefore, sequence diagrams may be examined to discover the states for an object (look at the space between the lines representing messages received by the object).
- ▶ If we had created sequence diagrams for each use case in the ESU Course Registration system, we would discover that objects in CourseOffering class can be in one of the following states:  
**Initialization** (created prior to registration but students have not been added to it), **Open** (able to accept students), **Closed** (maximum number of students already registered for it), **Canceled** (no longer offered).
- ▶ **CREATING STATES IN RATIONAL ROSE**
  1. Click to select the State icon from the toolbar.
  2. Click to place the state on the statechart diagram.
  3. With the state still selected, enter the name of the state.

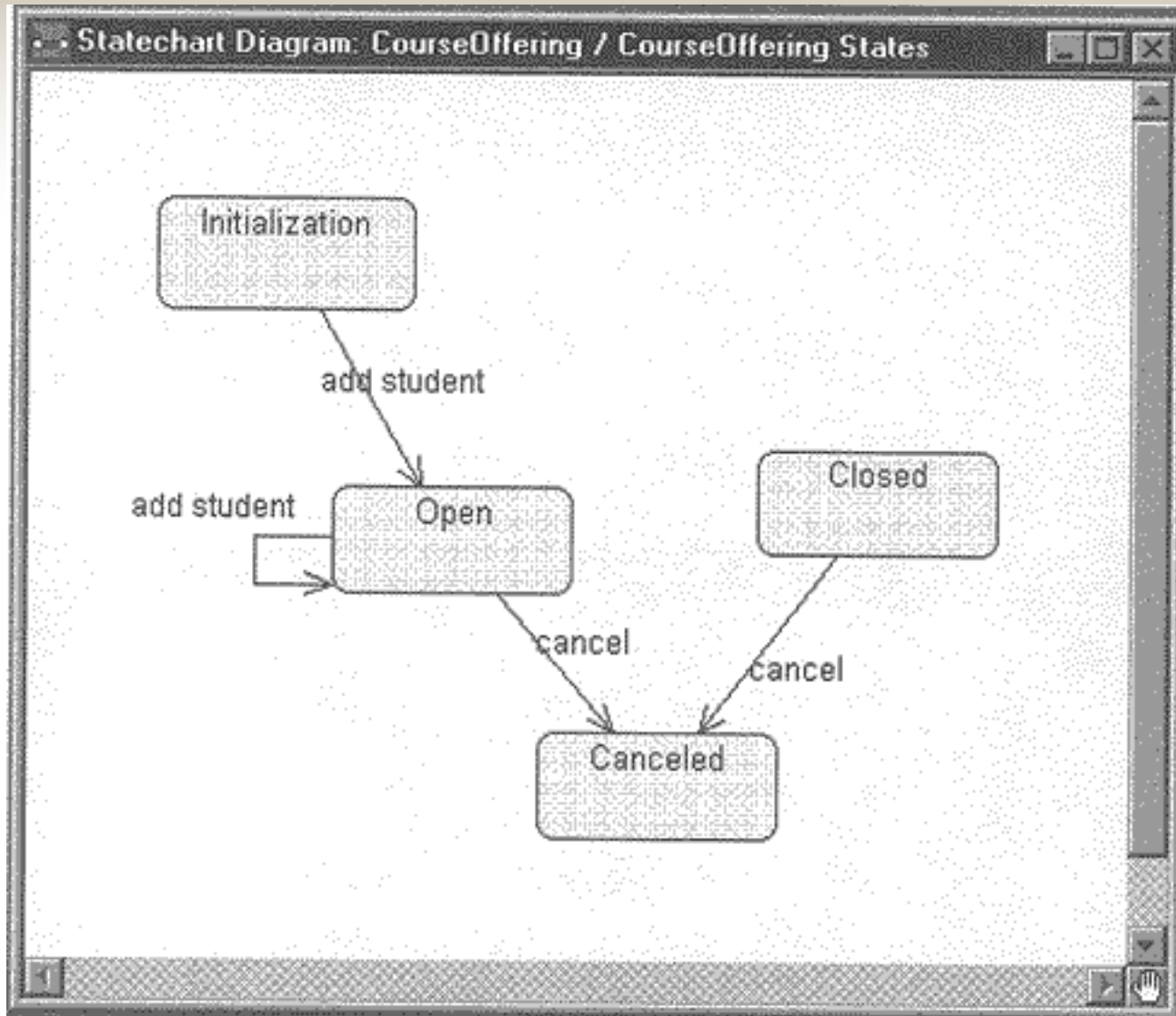
The states of the CourseOffering class are shown in Figure 9-3.



# State Transitions

- ▶ A state transition represents a change from an originating state to a successor state (which may be the same as the originating state). An action can accompany a state transition.
- ▶ There are two ways to transition out of a state—automatic and nonautomatic.
  - An automatic state transition occurs when the activity of the originating state completes—there is no named event associated with the state transition.
  - A nonautomatic state transition is caused by a named event (either from another object or from outside the system).
- ▶ Both types of state transitions are considered to take zero time and cannot be interrupted.
- ▶ A state transition is represented by an arrow that points from the originating state to the successor state.
- ▶ **CREATING STATE TRANSITIONS IN RATIONAL ROSE**
  1. Click to select the State Transition icon from the toolbar.
  2. Click to select the originating state on the statechart diagram.
  3. Drag the state transition to the successor state.
  4. If the state transition is a named transition, enter the name while the state transition arrow is still selected.

State transitions are shown in Figure 9-4.

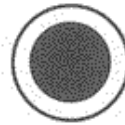


# Special States

- ▶ There are two special states that are added to the statechart diagram.
- ▶ The first is a start state. Each diagram must have one and only one start state since the object must be in a consistent state when it is created.
- ▶ The second special state is a stop state. An object can have multiple stop states. The



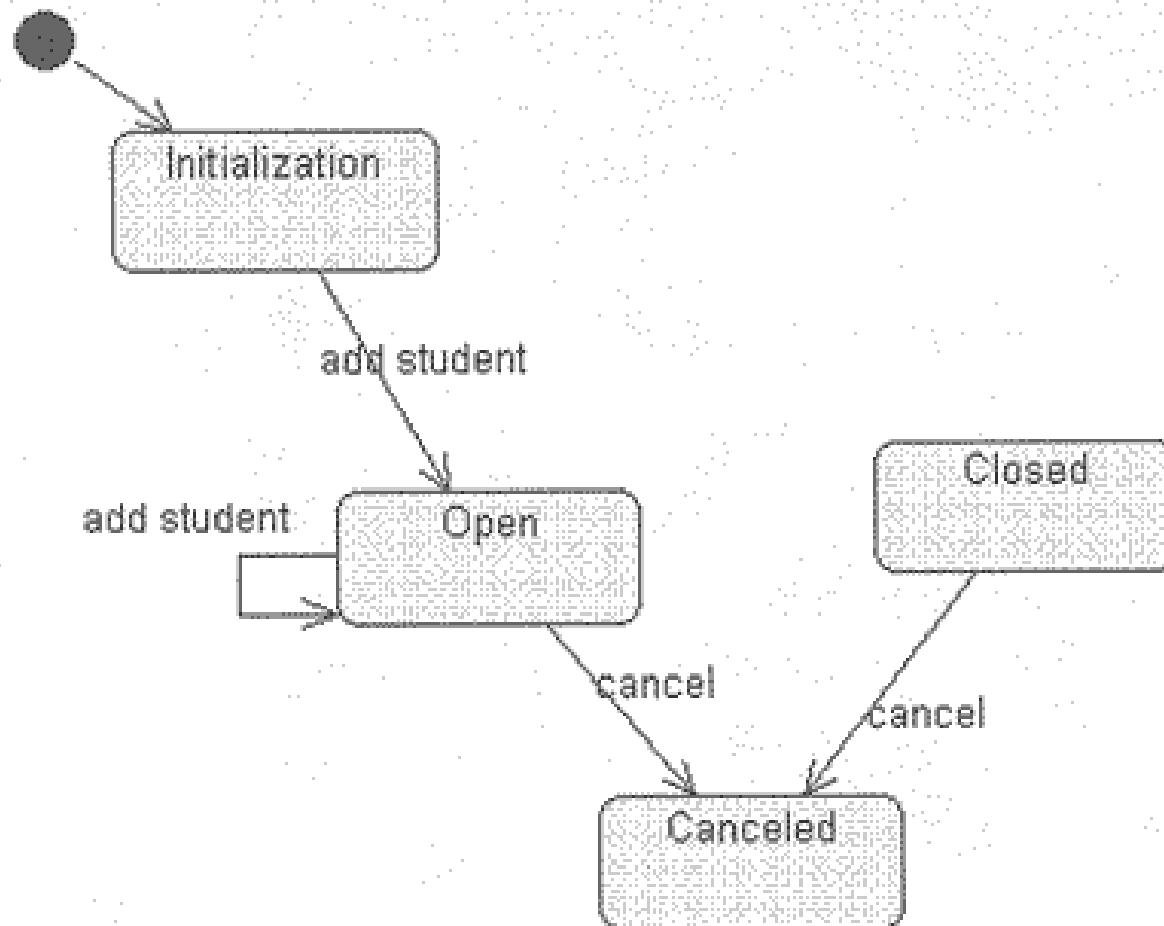
*Start State*



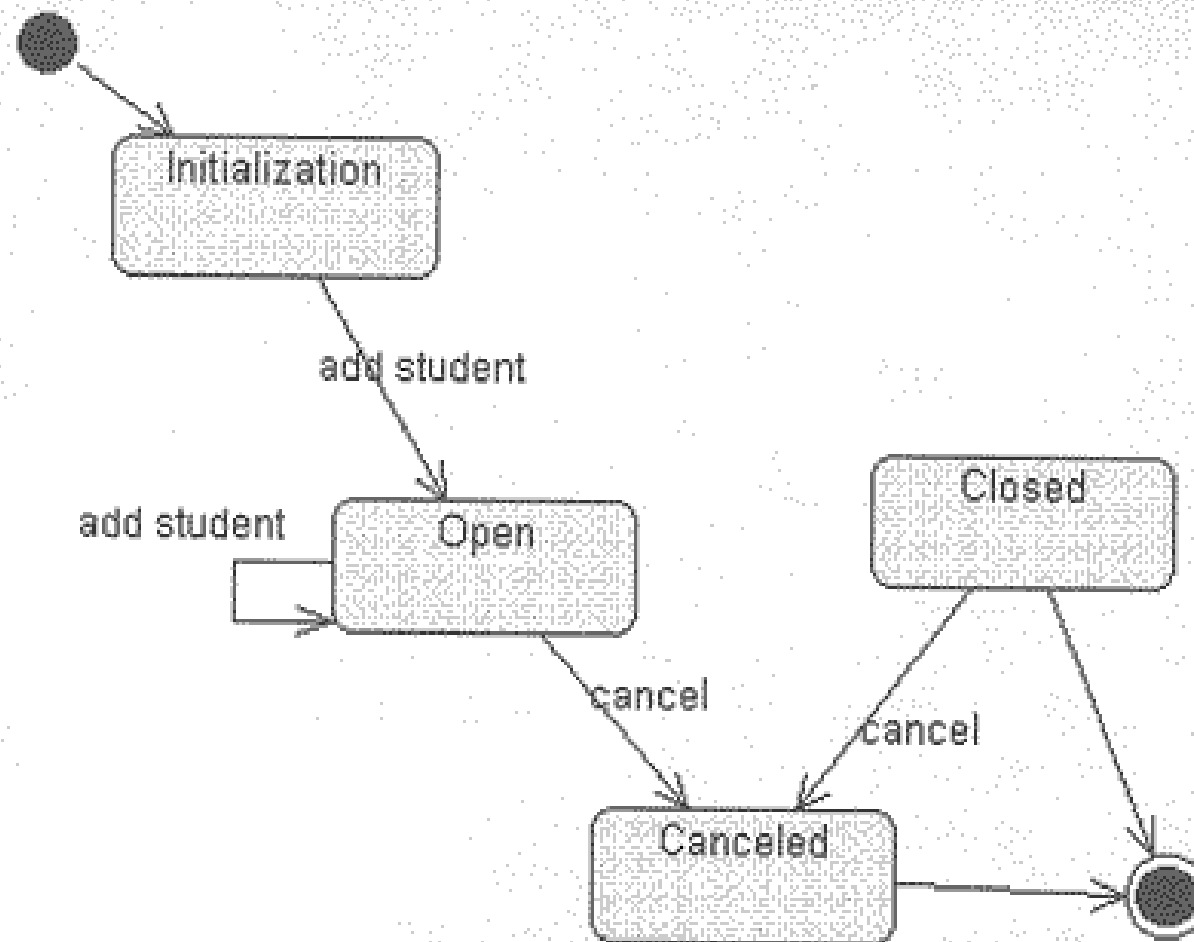
*Stop State*

- ▶ **CREATING START STATES IN RATIONAL ROSE**
  1. Click to select the Start icon from the toolbar.
  2. Click on the statechart diagram to draw the Start icon.
  3. Click to select the State Transition icon from the toolbar.
  4. Click on the Start icon and drag the arrow to the desired state.
- ▶ **CREATING STOP STATES IN RATIONAL ROSE**
  1. Select the Stop icon from the toolbar.
  2. Click on the statechart diagram to draw the Stop icon.
  3. Select the State Transition icon from the bar.
  4. Click on the state and drag the arrow to the Stop icon.

Statechart Diagram: CourseOffering / CourseOffering States



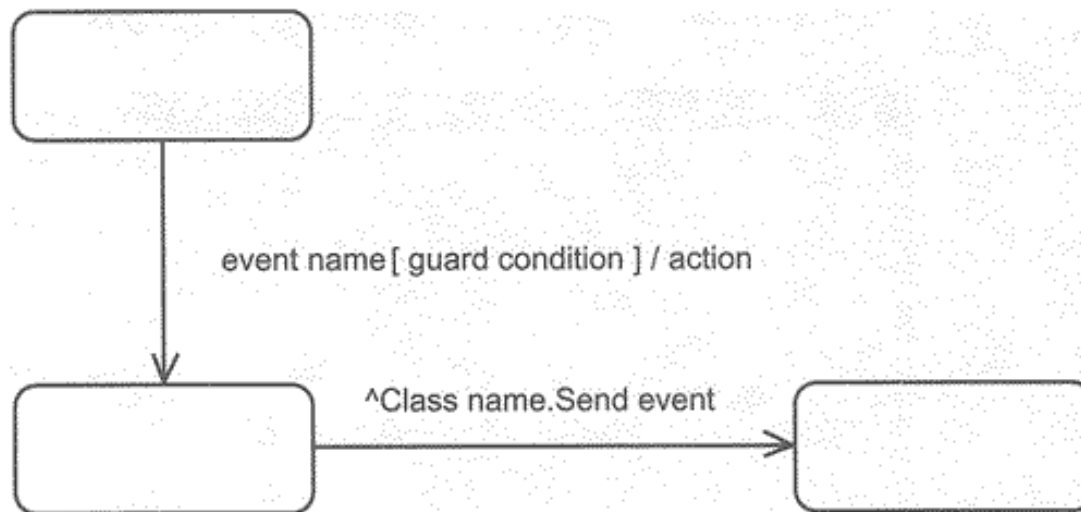
Statechart Diagram: CourseOffering / CourseOffering States





# State Transition Details

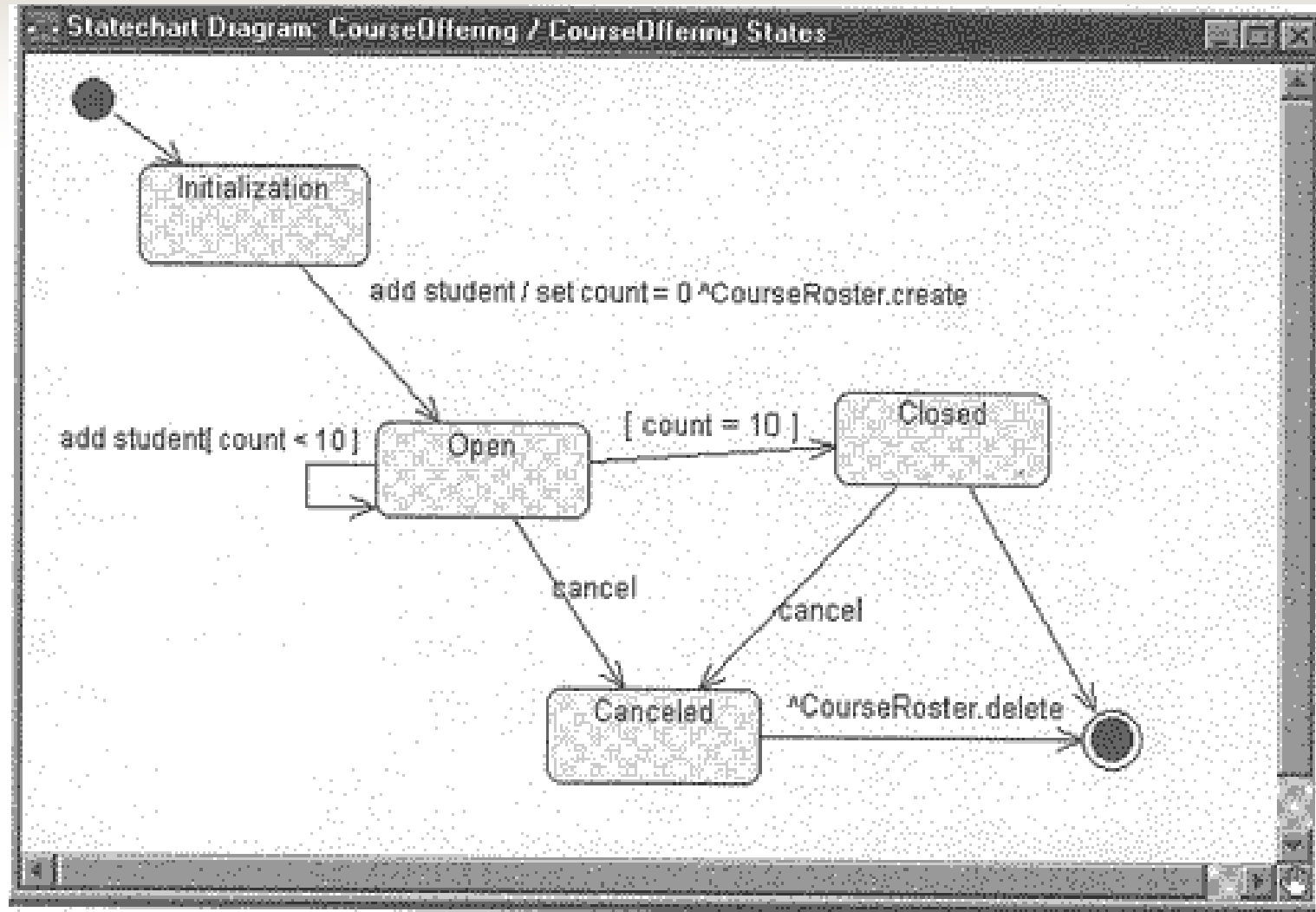
- ▶ A state transition may have an action and/or a guard condition associated with it and may also trigger an event.
- ▶ An action is behavior that occurs when the state transition occurs.
- ▶ An event is a message that is sent to another object in the system.
- ▶ A guard condition is a Boolean expression of attribute values that allows a state transition only if the condition is true.
- ▶ Both actions and guards are behaviors of the object and typically become operations. Often, these operations are private—that is, they are used only by the object itself.
- ▶ The UML notation for state transition detailed information is shown in Figure 9-8.



## ► ADDING STATE TRANSITION DETAILS IN RATIONAL ROSE

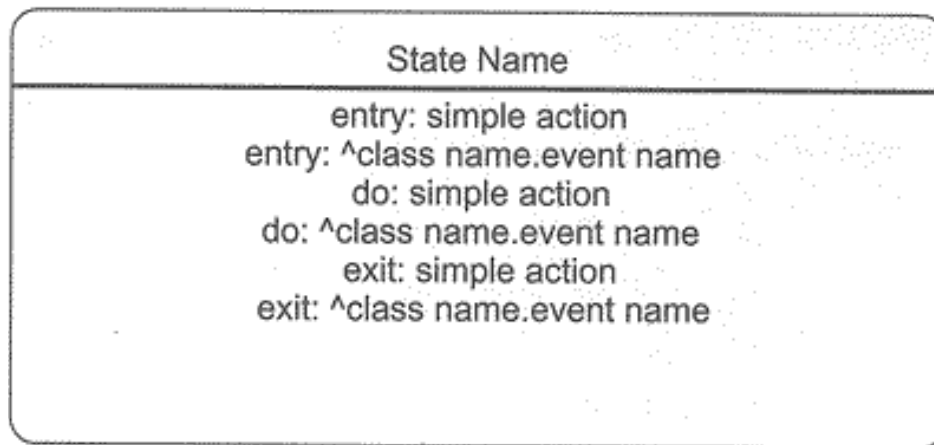
1. Right-click on the state transition arrow to make the shortcut menu visible.
2. Select the Open Specification menu choice.
3. Select the Detail tab.
4. Enter the action, guard, and/or the event to be sent.
5. Click the OK button to close the specification.

State transition detail is shown in Figure 9-9.



# State Details

- ▶ Actions that accompany all state transitions into a state may be placed as an entry action within the state.
- ▶ Likewise, actions that accompany all state transitions out of a state may be placed as exit actions within the state.
- ▶ Behavior that occurs within the state is called an activity.
- ▶ An activity starts when the state is entered and either completes or is interrupted by an outgoing state transition.
- ▶ The behavior may be a simple action or it may be an event sent to another object.
- ▶ As with actions and guards, this behavior typically is mapped to operations on the object.
- ▶ The UML notation for state detailed information is shown in Figure 9-10.



# CREATING ENTRY ACTIONS, EXIT ACTIONS, AND ACTIVITIES IN RATIONAL ROSE

1. Right-click on the state to make the shortcut menu visible.
2. Select the Open Specification menu choice.
3. Select the Actions tab.
4. Right-click in the Action field to make the shortcut menu visible.
5. Select the Insert menu choice. This will create an action called entry.
6. Double-click on entry to make the Action Specification visible.
7. Select when the action should occur: on entry, on exit, do, or on event.
8. Enter the action or event information.
9. Select the type: action or send event.
10. Enter the action name and event information (if needed).
11. Click the OK button to close the Action Specification.
12. Click the OK button to close the State Specification.

# Figure 9-10. State Details

