Chapter 10 Describe the Run-time Architecture

Agenda

- Objectives
- Context in the Run-time Architecture
- Describe the Run-time Architecture Steps
- Exercises

Objectives: Describe the Run-time Architecture

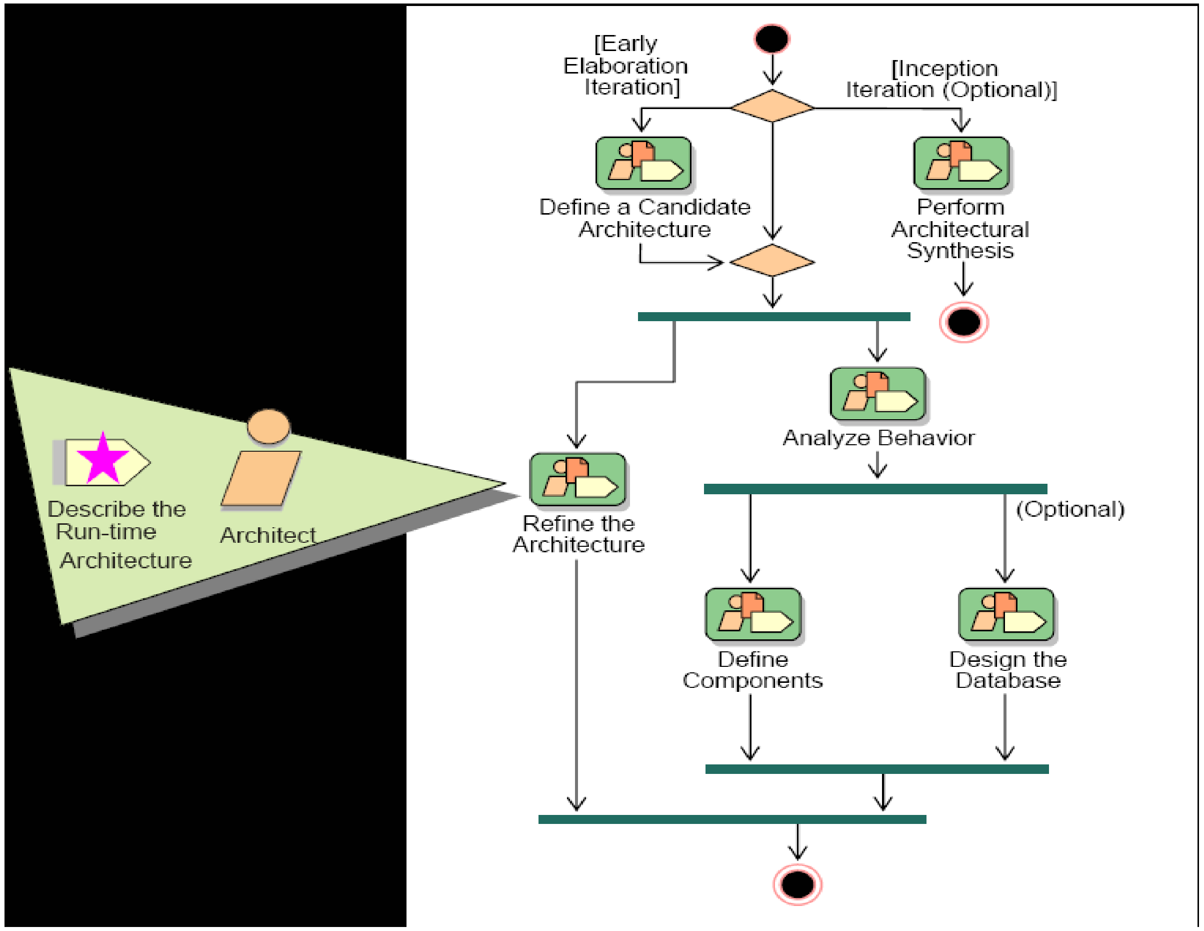
- **Define the purpose of the Describe the Run-time Architecture activity and when in the lifecycle it is performed**
- **Demonstrate how to model processes and threads**
- **Explain how to model what classes and subsystems are mapped to processes and threads**
- **Define the rationale and considerations that support architectural decisions**

Chapter 10 Describe the Run-time Architecture

Agenda

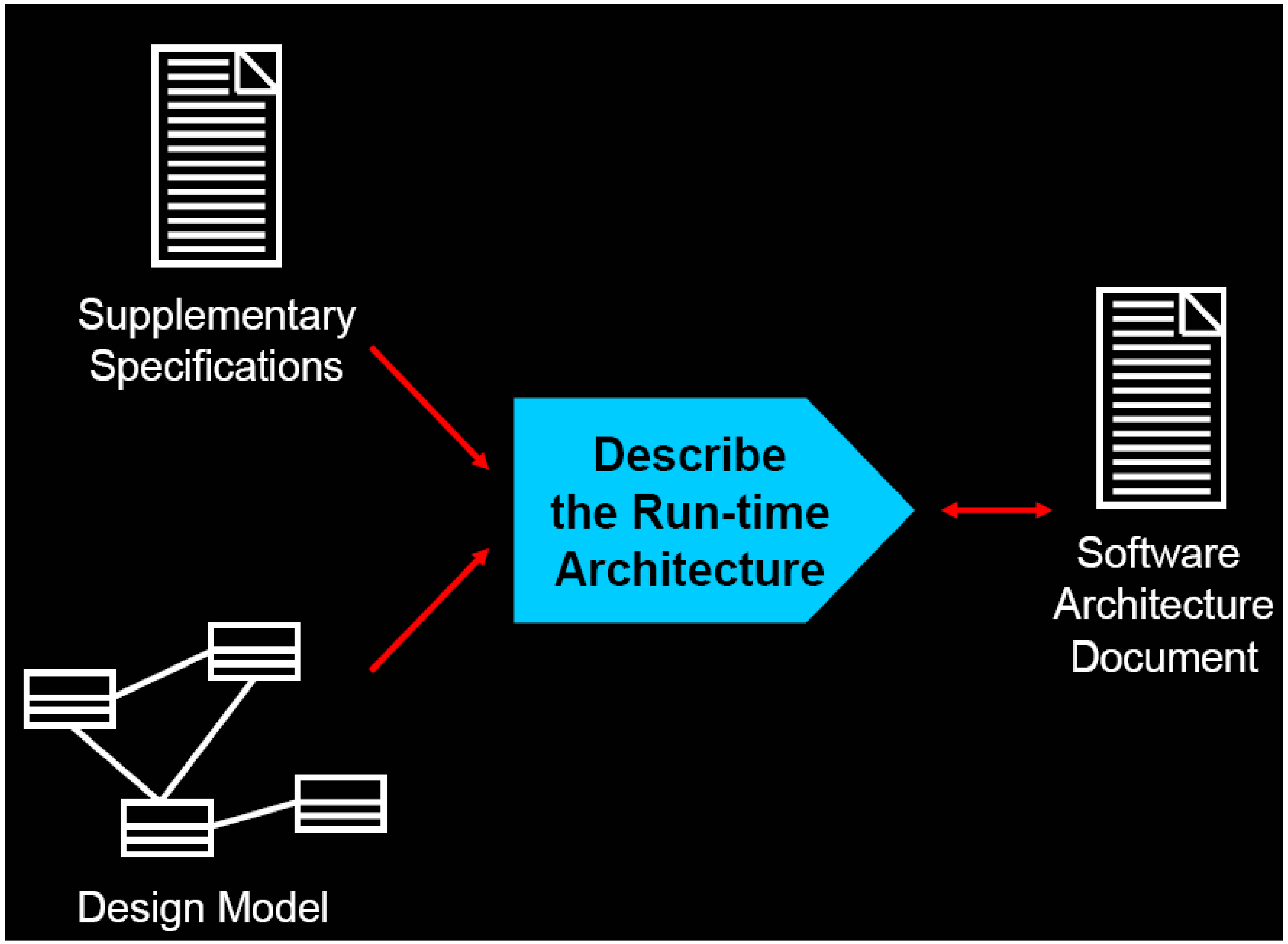
- Objectives
- Context in the Run-time Architecture
- Describe the Run-time Architecture Steps
- Exercises

Describe the Run-time Architecture in Context

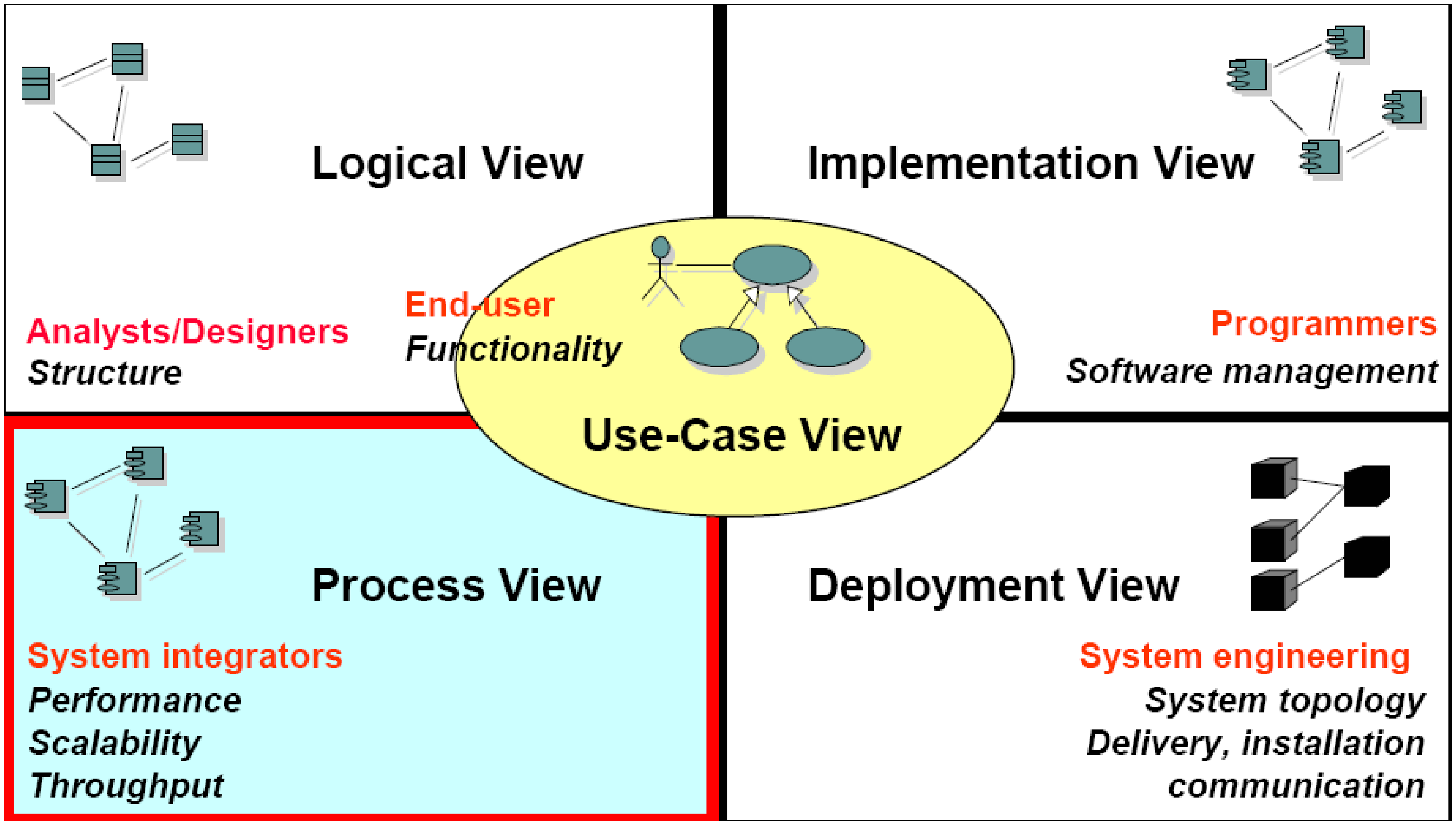

[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Describe the Run-time Architecture
Architect
Refine the Architecture
(Optional)
Define Components
Design the Database

2020/9/10

Describe the Run-time Architecture Overview


Supplementary Specifications
Design Model
Describe the Run-time Architecture
Software Architecture Document

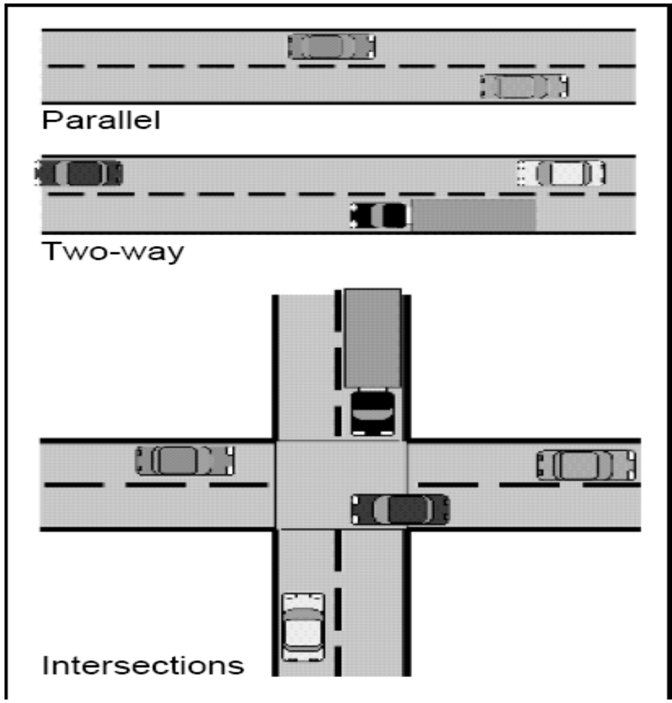
Key Concepts: The Process View



The Process View is an "architecturally significant" slice of the processes and threads of the Design Model

What Is Concurrency?

- **Example of concurrency at work:**
  - **Parallel roads require little coordination**
  - **Two-way roads require some coordination for safe interaction**
  - **Intersections require careful coordination**

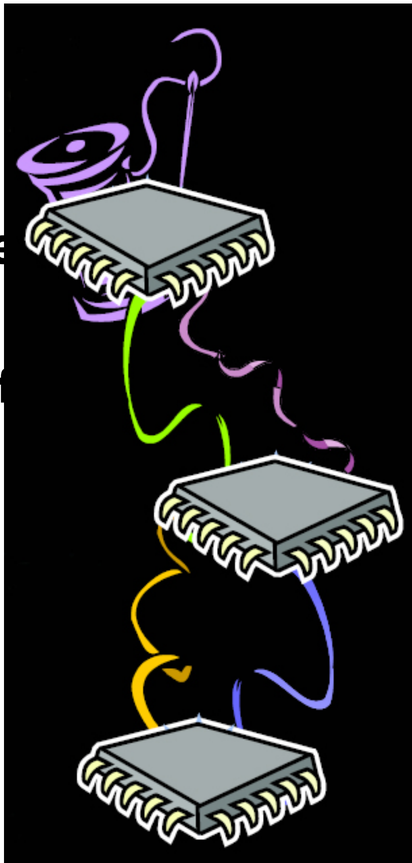


Why Are We Interested in Concurrency?

- Software might need to respond to seemingly random externally generated events
- Performing tasks in parallel can improve performance if multiple CPUs are available
  - Example: Startup of a system
- Control of the system can be enhanced through concurrency

Realizing Concurrency: Concurrency Mechanisms

- **To support concurrency, a system must provide for multiple threads of control**
- **Common concurrency mechanisms**
  - **Multiprocessing**
    - ◆ **Multiple CPUs execute concurrently**
  - **Multitasking**
    - ◆ **The operating systems simulate concurrency CPU by interleaving the execution of differe**
  - **Application-based solutions**
    - ◆ **the application software takes responsibility f switching between different branches of code at appropriate times**



Chapter 10 Describe the Run-time Architecture

Agenda

- **Objectives**
- **Context in the Run-time Architecture**
- **Describe the Run-time Architecture Steps**
- **Exercises**

Describe the Run-time Architecture Steps

- **Analyze concurrency requirements**
- **Identify processes and threads**
- **Identify process lifecycles**
- **Map processes onto the implementation**
- **Distribute model elements among processes**

Concurrency Requirements

- Concurrency requirements are driven by:
  - The degree to which the system must be distributed.
  - The degree to which the system is event driven.
  - The computation intensity of key algorithms.
  - The degree of parallel execution supported by the environment
- Concurrency requirements are ranked in terms of importance to resolve conflicts.

Example: Concurrency Requirements

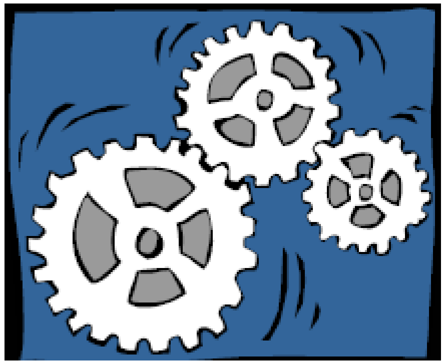
- In the Course Registration System, the concurrency requirements come from the requirements and the architecture:
  - Multiple users must be able to perform their work concurrently
  - If a course offering becomes full while a student is building a schedule including that offering, the student must be notified
  - Risk-based prototypes have found that the legacy course catalog database cannot meet our performance needs without some creative use of mid-tier processing power

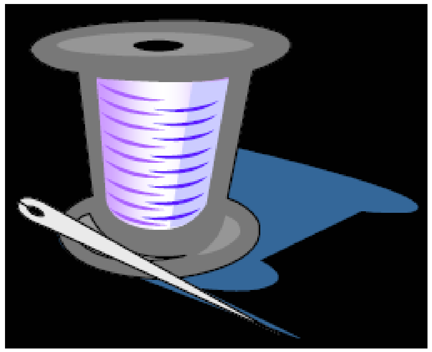
Describe the Run-time Architecture Steps

- **Analyze concurrency requirements**
- **Identify processes and threads**
- **Identify process lifecycles**
- **Map processes onto the implementation**
- **Distribute model elements among processes**

Key Concepts: Process and Thread

- **Process: Provides heavyweight flow of control**
  - **Is stand-alone**
  - **Can be divided into individual threads**
- **Thread: Provides lightweight flow of control**
  - **Runs in the context of an enclosing process**



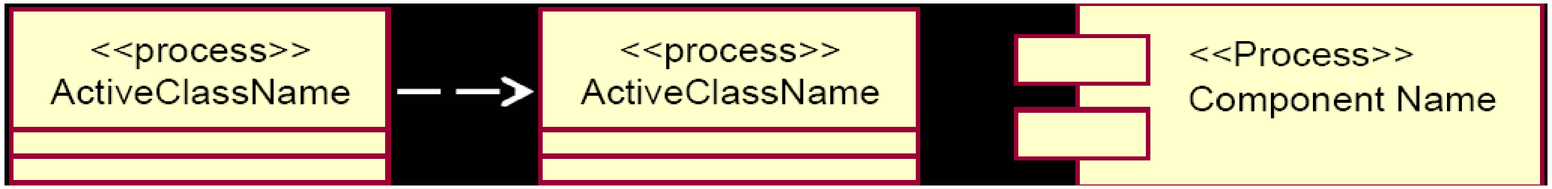


Identifying Processes and Threads

- For each separate flow of control needed by the system, create a process or thread
  - Separate threads of control might be needed to:
    - ◆ Utilize multiple CPUs and/or nodes
    - ◆ Increase CPU utilization
    - ◆ Service time-related events
    - ◆ Prioritize activities
    - ◆ Achieve scalability (load sharing)
    - ◆ Separate the concerns among software a
    - ◆ Improvement of system availability
    - ◆ Support major subsystems

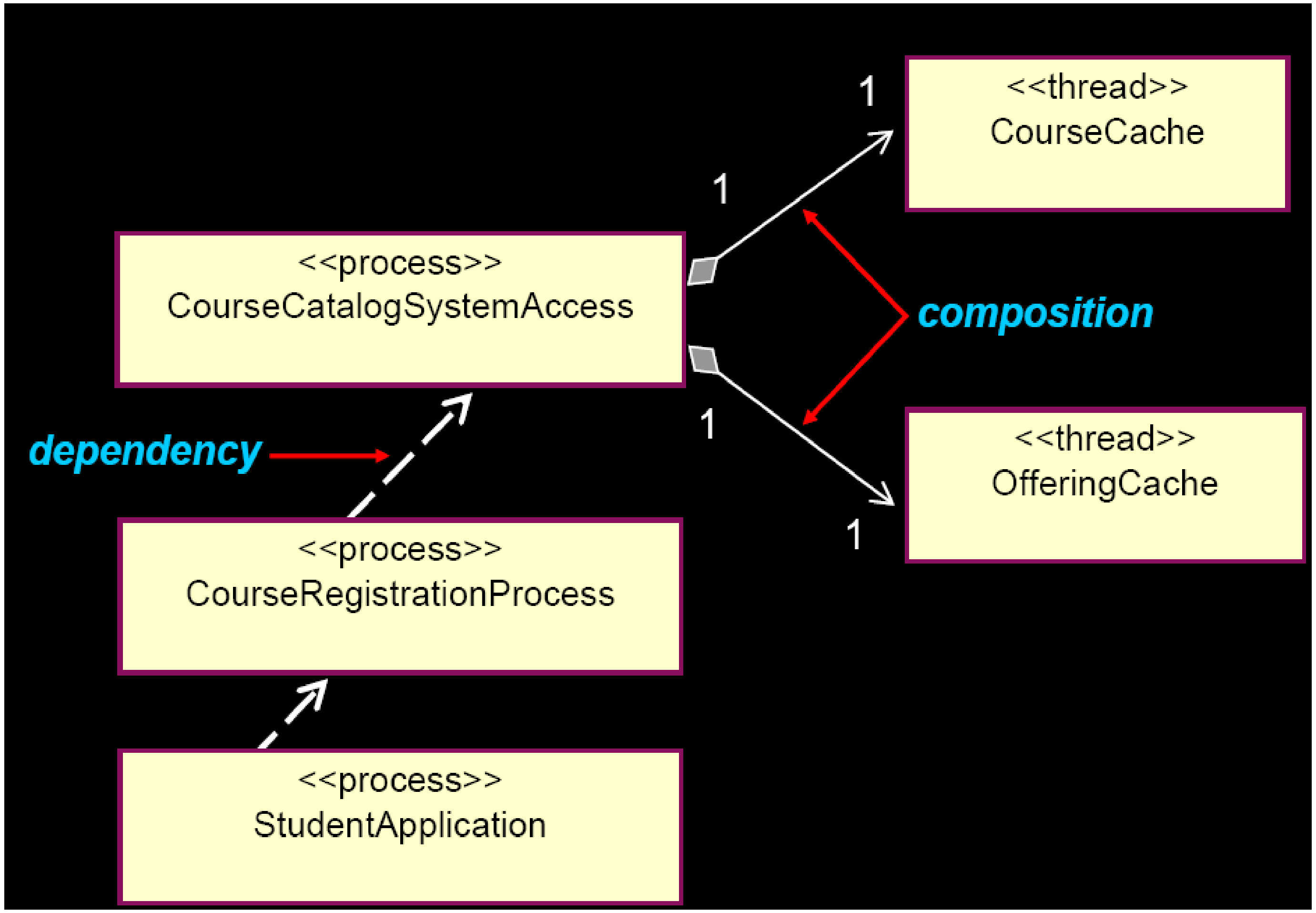
Modeling Processes

- Processes can be modeled using
  - Active classes (Class Diagrams) and Objects (Interaction Diagrams)
  - Components (Component Diagrams) Stereotypes: <<process>> or <<thread>>
- Process relationships can be modeled as dependencies

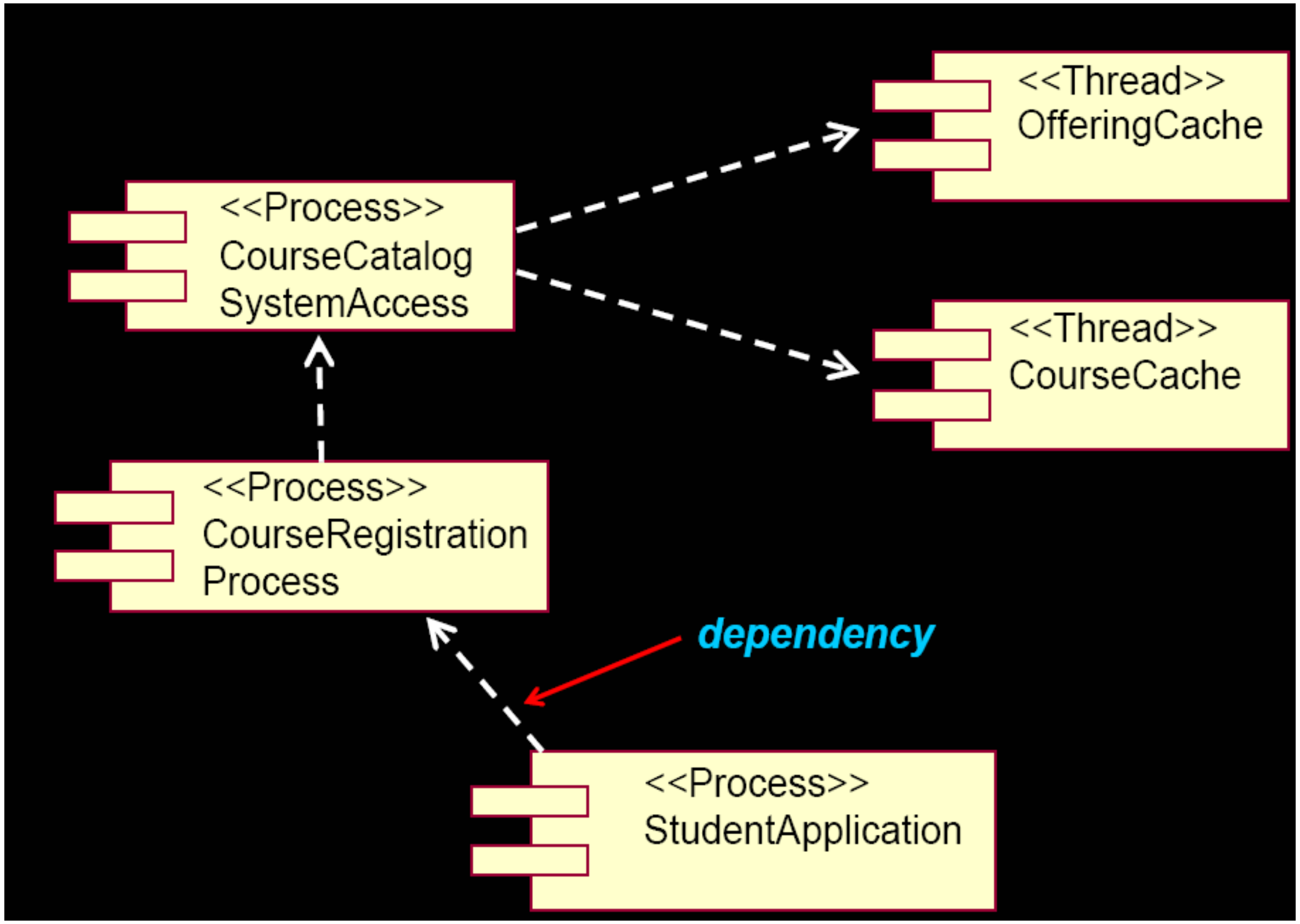


This course will model processes and threads using Class Diagrams.

Example: Modeling Processes: Class Diagram



Example: Modeling Processes: Component Diagram



Describe the Run-time Architecture Steps

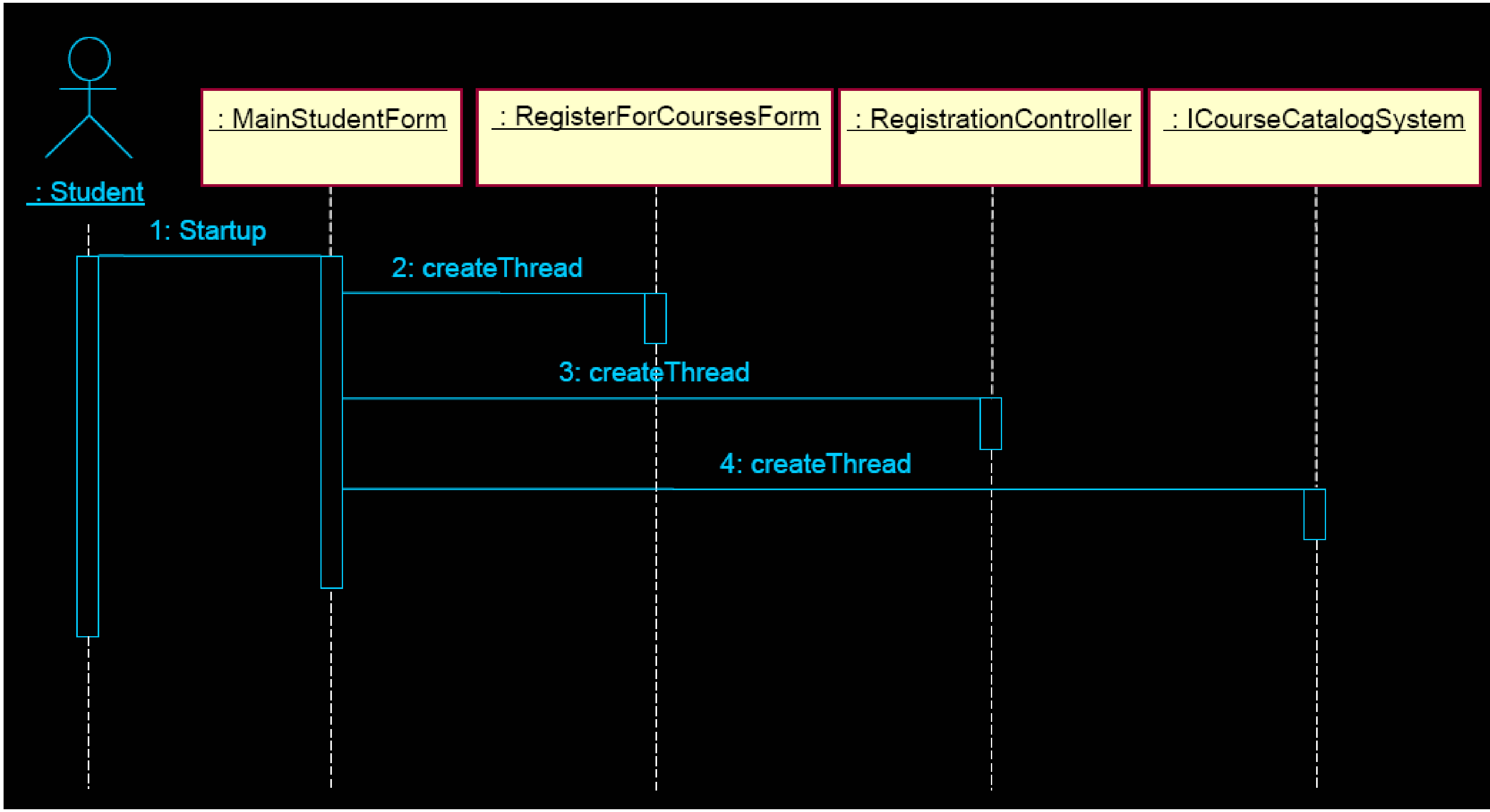
- **Analyze concurrency requirements**
- **Identify processes and threads**
- **Identify process lifecycles**
- **Map processes onto the implementation**
- **Distribute model elements among processes**

Creating and Destroying Processes and Threads

- Single-process architecture
  - □– Process creation takes place when the application starts
  - □– Process destruction takes place when the application ends
- Multi-process architecture
  - □– New processes are typically created from the initial process that was created when the application was started
  - □– Each process must be individually destroyed

Note: The Course Registration System utilizes a multi-process architecture

Example: Create Processes and Threads



Creation of threads during application startup

Describe the Run-time Architecture Steps

- **Analyze concurrency requirements**
- **Identify processes and threads**
- **Identify process lifecycles**
- **Map processes onto the implementation**
- **Distribute model elements among processes**

Mapping Processes onto the Implementation

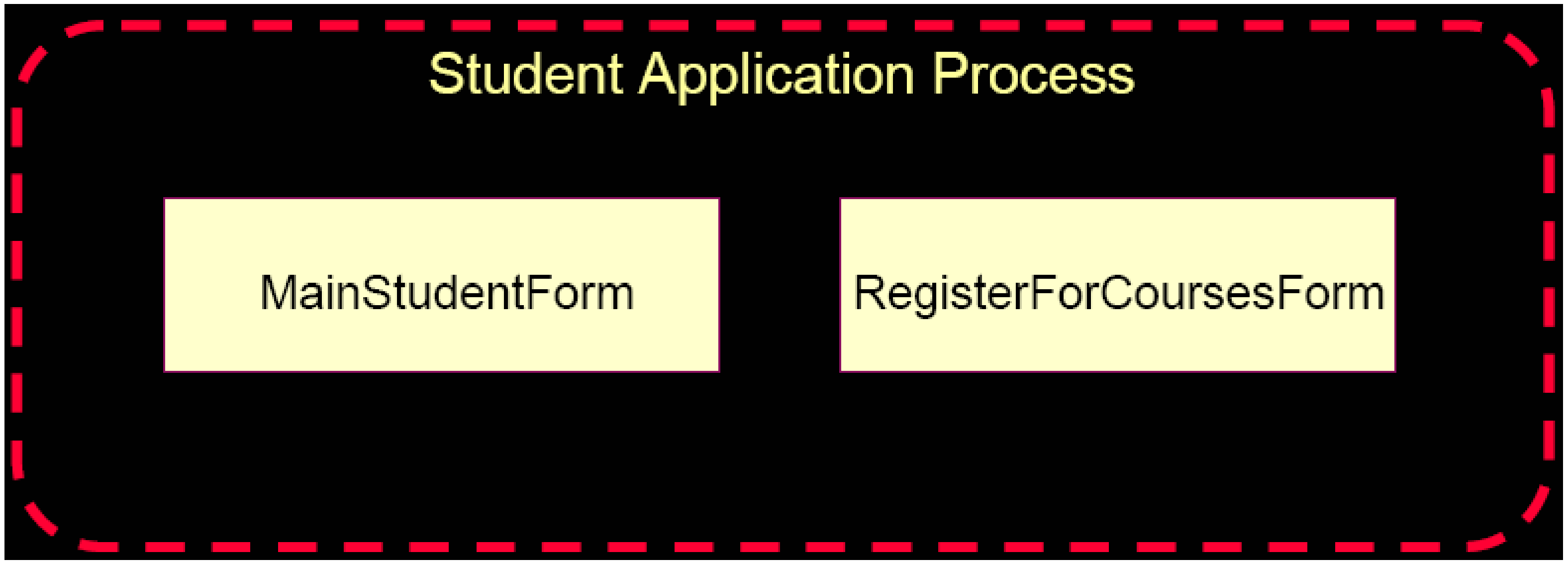
- **Processes and threads must be mapped onto specific implementation constructs**
- **Considerations**
  - □ – **Process coupling**
  - □ – **Performance requirements**
  - □ – **System process and thread limits**
  - □ – **Existing threads and processes**
  - □ – **IPC (电路资源) resource availability**

Describe the Run-time Architecture Steps

- **Analyze concurrency requirements**
- **Identify processes and threads**
- **Identify process lifecycles**
- **Map processes onto the implementation**
- **Distribute model elements among processes**

Design Element Allocation

- Instances of a given class or subsystem *must* execute within at least one process
  – They may execute in several processes



Design Elements-to-Processes Considerations

- **Based on:**
  - □ – Performance and concurrency requirements
  - □ – Distribution requirements and support for parallel execution
  - □ – Redundancy and availability requirements
- **Class/subsystem characteristics to consider:**
  - □ – Autonomy
  - □ – Subordination
  - □ – Persistence
  - □ – Distribution

数据持久（1）

- 持久数据存储就是即使在服务器崩溃的情况下仍能存在的数据存储
- 美国国家标准与技术研究所（The United States National Institute of Standards and Technology）定义了三种级别的持久数据：
  - 部分持久数据是一种仅允许对最新版本更新的持久数据结构
  - 持久数据是一种保留其旧版本的数据结构；即，以前版本和当前版本都可能被查询
  - 完全持久数据是一种维护其数据的所有版本并允许对这些版本更新的持久数据结构

数据持久（II）

- 大多数业务应用程序至少提供部分持久数据。这种类型的持久性在事务中期或者甚至在请求中期出现系统故障时容易遭破坏，这会导致数据不完整且常常遭毁坏
- 另一方面，在持久数据实现中，对系统中断或故障以“回滚（rollback）”回应，数据状态被回滚到上一个已知的良好配置
- 持久数据实现在企业体系结构和数据库管理系统（DBMS）中很常见
- 完全持久数据实现非常少见。完全持久数据实现的少数几个示例有：日志记录文件系统、VMS 文件系统（如 VAX 和 Mac OS X）以及并发版本控制系统（CVS）

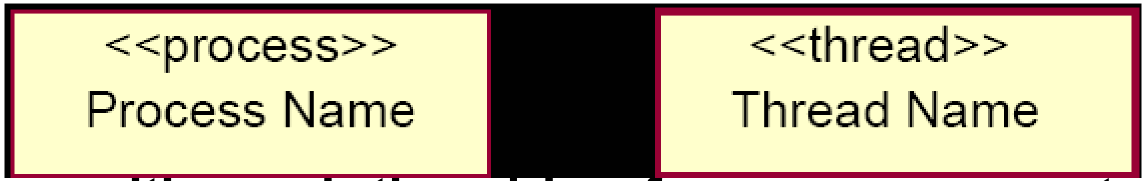
Design Elements-to-Processes Strategies

Two Strategies (used simultaneously)

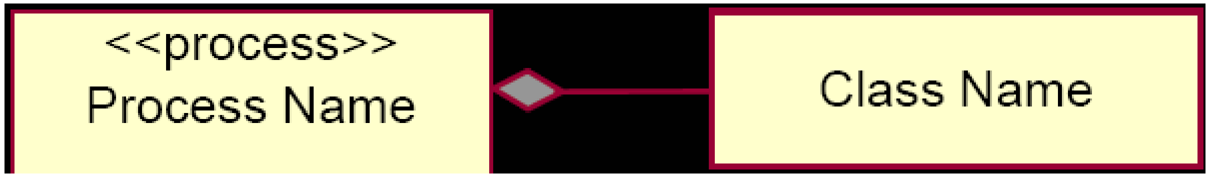
- **Inside-Out**
  - □ – **Group elements that closely cooperate and must execute in the same thread of control**
  - □ – **Separate elements that do not interact**
  - □ – **Repeat until you reach the minimum number of processes that still provide the required distribution and effective resource utilization**
- **Outside-In**
  - □ – **Define a separate thread of control for each external stimuli**
  - □ – **Define a separate server thread of control for each service**
  - □ – **Reduce number of threads to what can be supported**

Modeling the Mapping of Elements to Processes

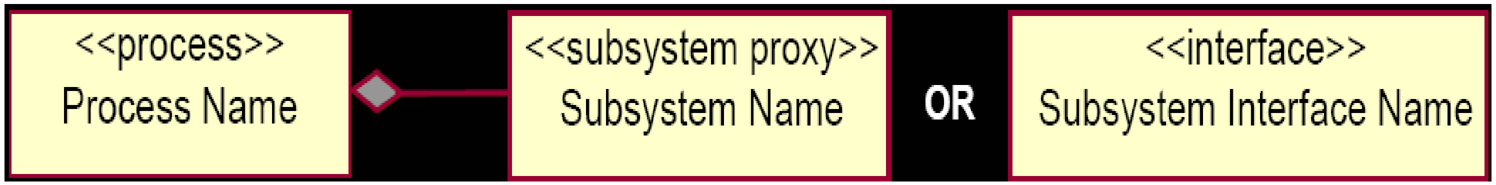
- Class diagrams
  - Active classes as processes



  - Composition relationships from processes to classes

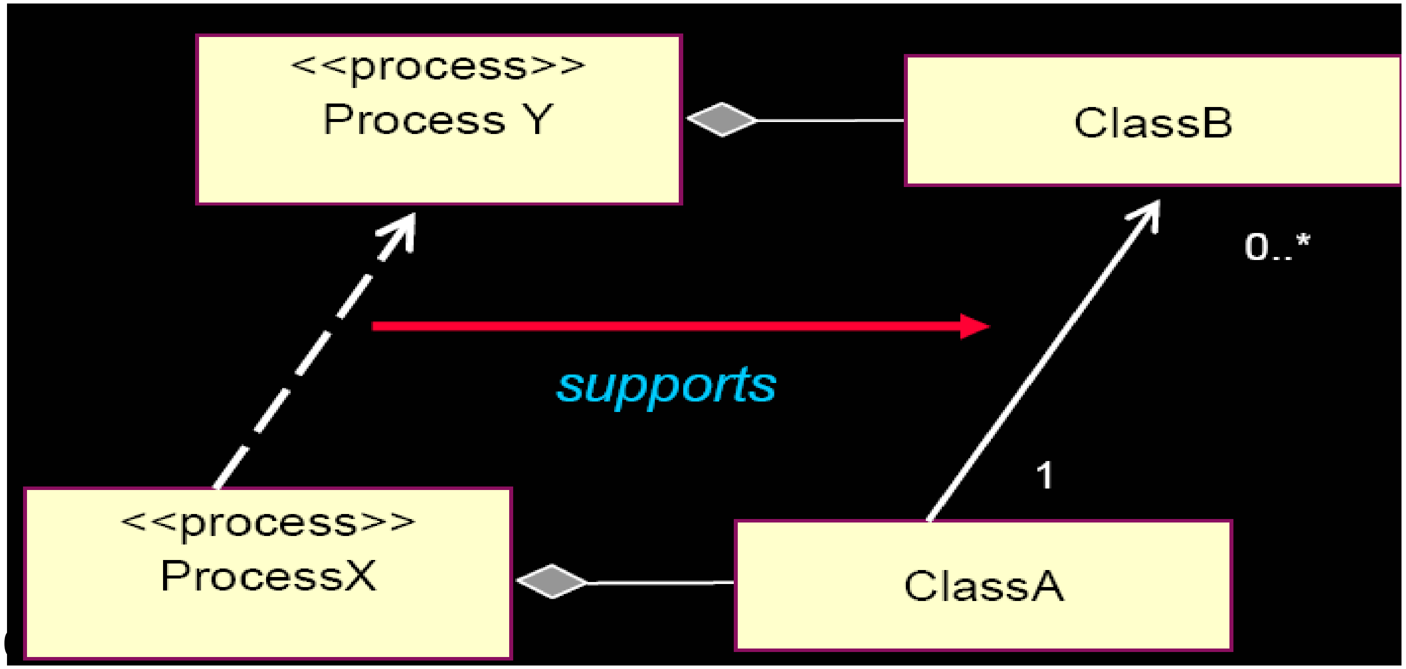


  - Composition relationships from processes to subsystems

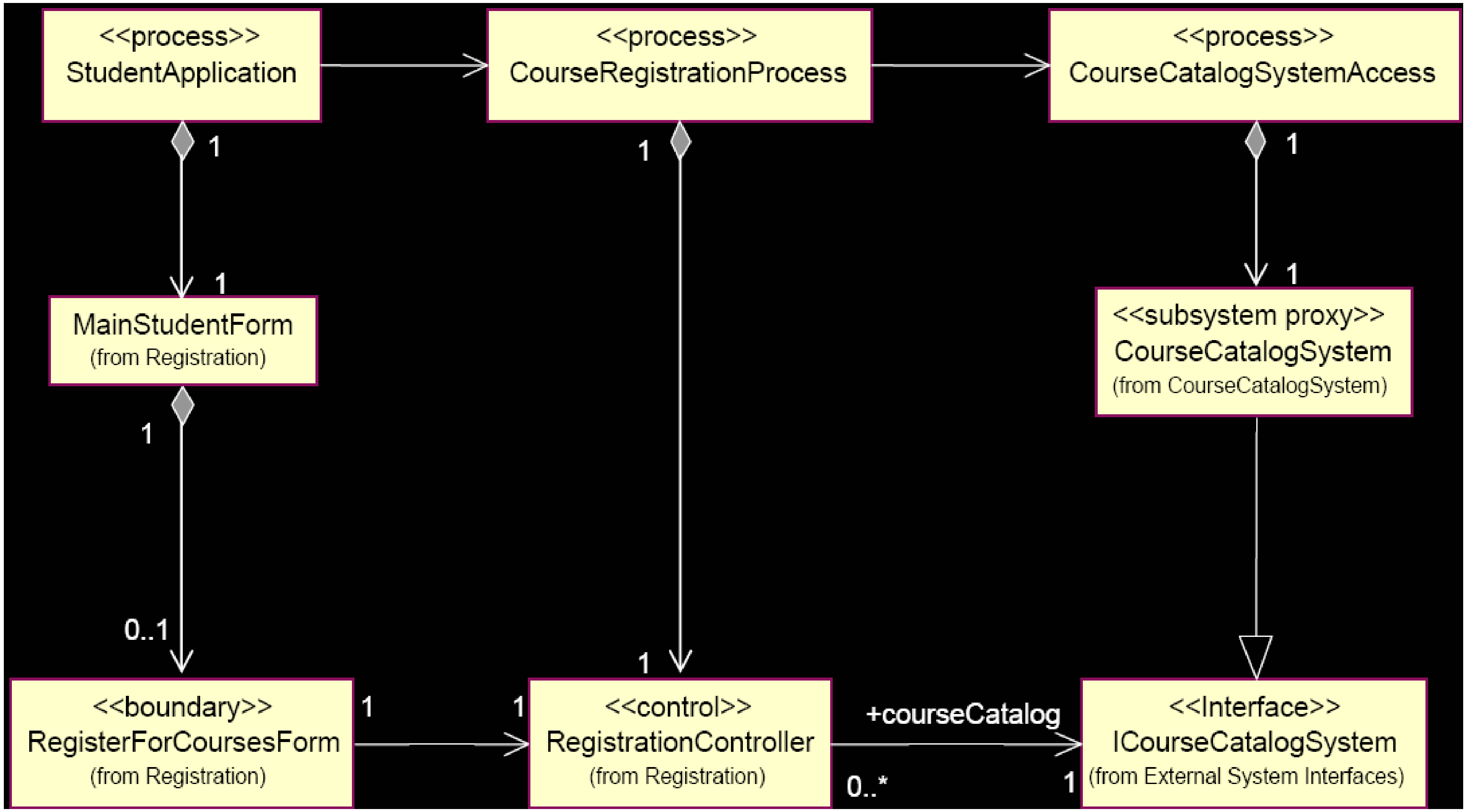


Process Relationships

- **Process relationships must support design element relationships**

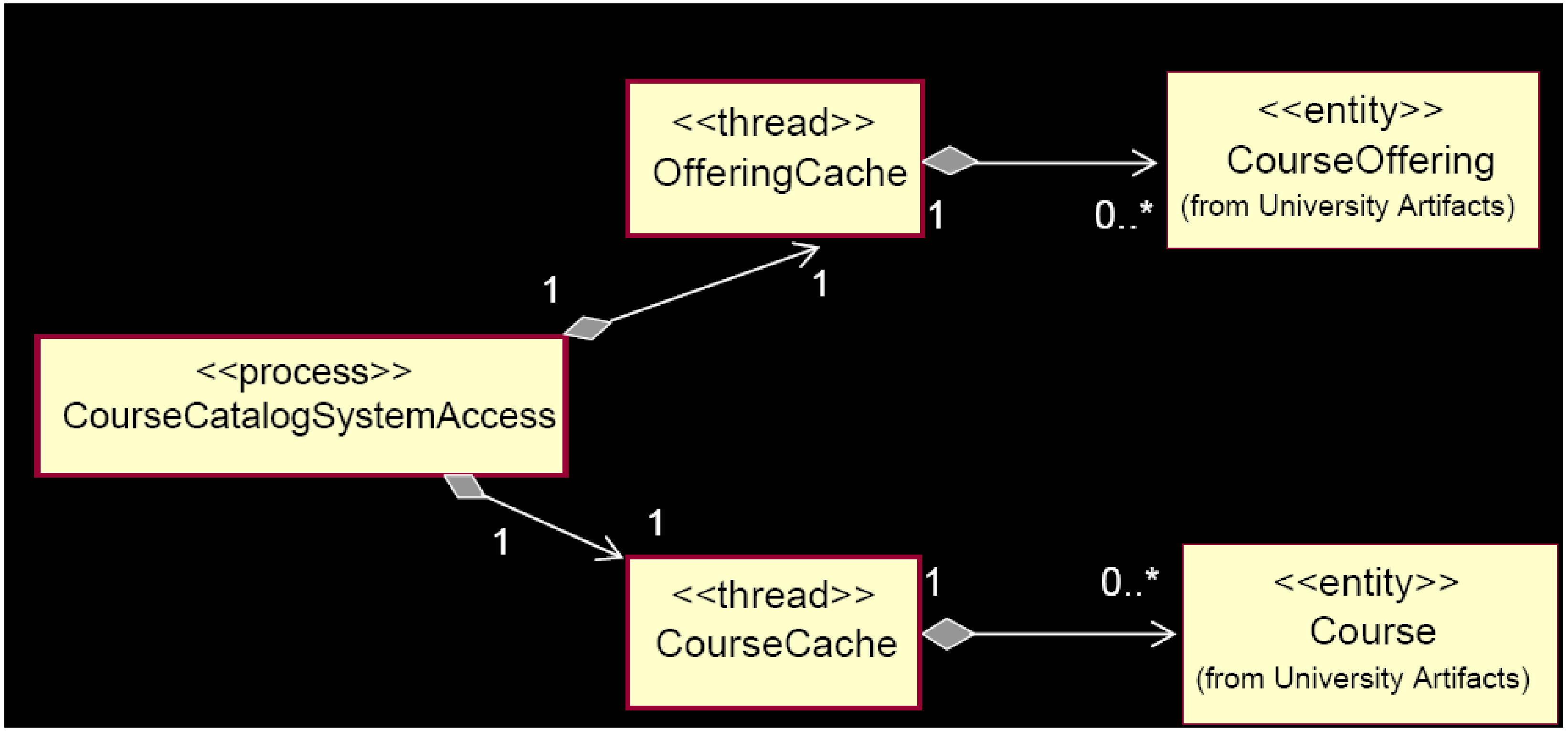


Example: Register for Course Processes



2020/9/10

Example: Register for Course Processes (cont.)



Checkpoints: Describe the Run-time Architecture

- **What is the purpose of the Describe Concurrency activity?**
- **What is a process? What is a thread?**
- **Describe some of the considerations when identifying processes.**
- **Describe the two strategies for mapping classes and subsystems to processes.**
- **How do you model the Process View ?**
- **What modeling elements and diagrams are used?**

作业

1. **What is the purpose of the Describe the Run-time Architecture activity?**
2. **What is a process? What is a thread?**
3. **Describe some of the considerations when identifying processes.**
4. **Describe the two strategies for mapping classes and subsystems to processes.**
5. **What modeling elements and diagrams are used?**

作业 10 th

1. 系统部件设计（系统运行时架构设计）的意义？
2. 何为进程？何为线程？
3. 在进行部件设计时，应考虑哪些因素？（其实就是怎么做）
4. 进行部件设计后，将其纳入系统实现方案时，还应考虑哪些因素？（部件设计之后，映射到类、子系统等）
5. 在进行基于面向对象的部件设计，尤其是设计系统运行架构时，可以用UML的哪些图？

Chapter 10 Describe the Run-time Architecture

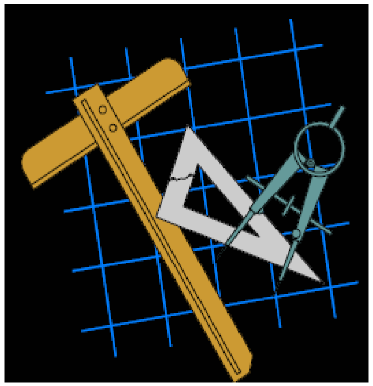
Agenda

- Objectives
- Context the Run-time Architecture
- Describe the Run-time Architecture Steps
- Exercises

Lab: Describe the Run-time Architecture

- Given the following:
  - □– Design elements (classes and subsystems) and their relationships Processes（类、子系统及其接口）

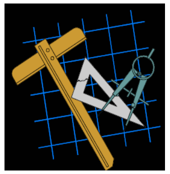
Exercise: Describe the Run-time Architecture



- **Produce the following:**
  - **Class diagram showing the:**
    - **Processes**
    - **Mapping of classes and subsystems to processes**
    - **Process relationships**
    - **Design element relationships to support process relationships**

Exercise: Describe the Run-time Architecture

- 完成如下制品:
  - 用文字描述发生并发需求的情况和级别
  - 用类图描述出解决上述并发的方案，可以用**进程、线程或构件**
  - 定义相关进程、线程或构件的生命周期
  - 将上述部件设计纳入系统的类的设计之中，**提供必要的文字描述或UML图**



Lab: Review

- Compare your Process View with those created by the rest of the class
  - Are processes and threads stereotyped properly?
    If a thread is defined, is there a composition relationship from the process to the thread?
  - Is there a composition relationship from the process elements to the design elements?
  - Do the necessary relationships exist between the process elements in order to support the relationships
    to the design elements mapped to those process elements?