IBM Software Group

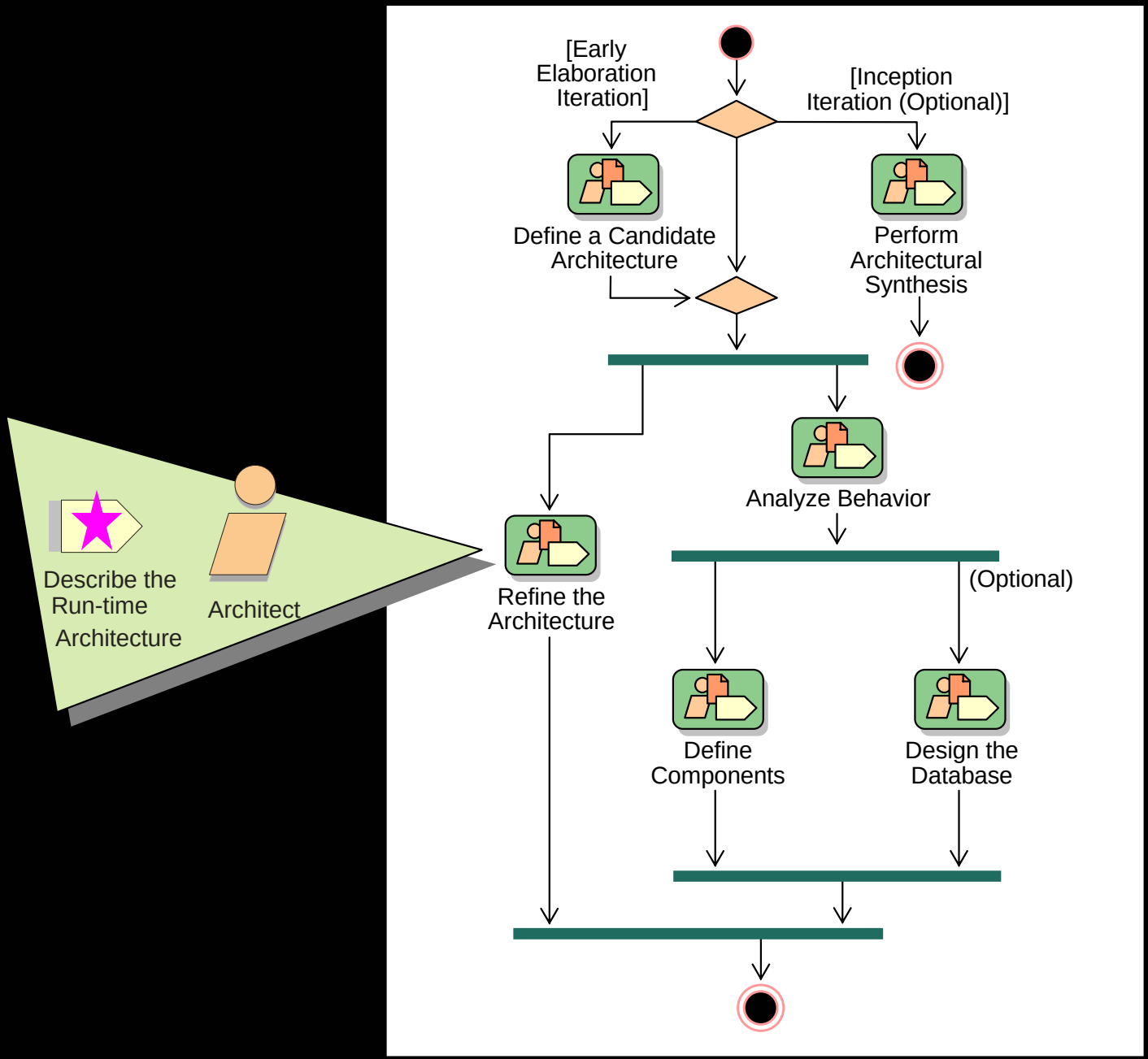
Mastering Object-Oriented Analysis and Design with UML

Module 7: Describe the Run-time Architecture

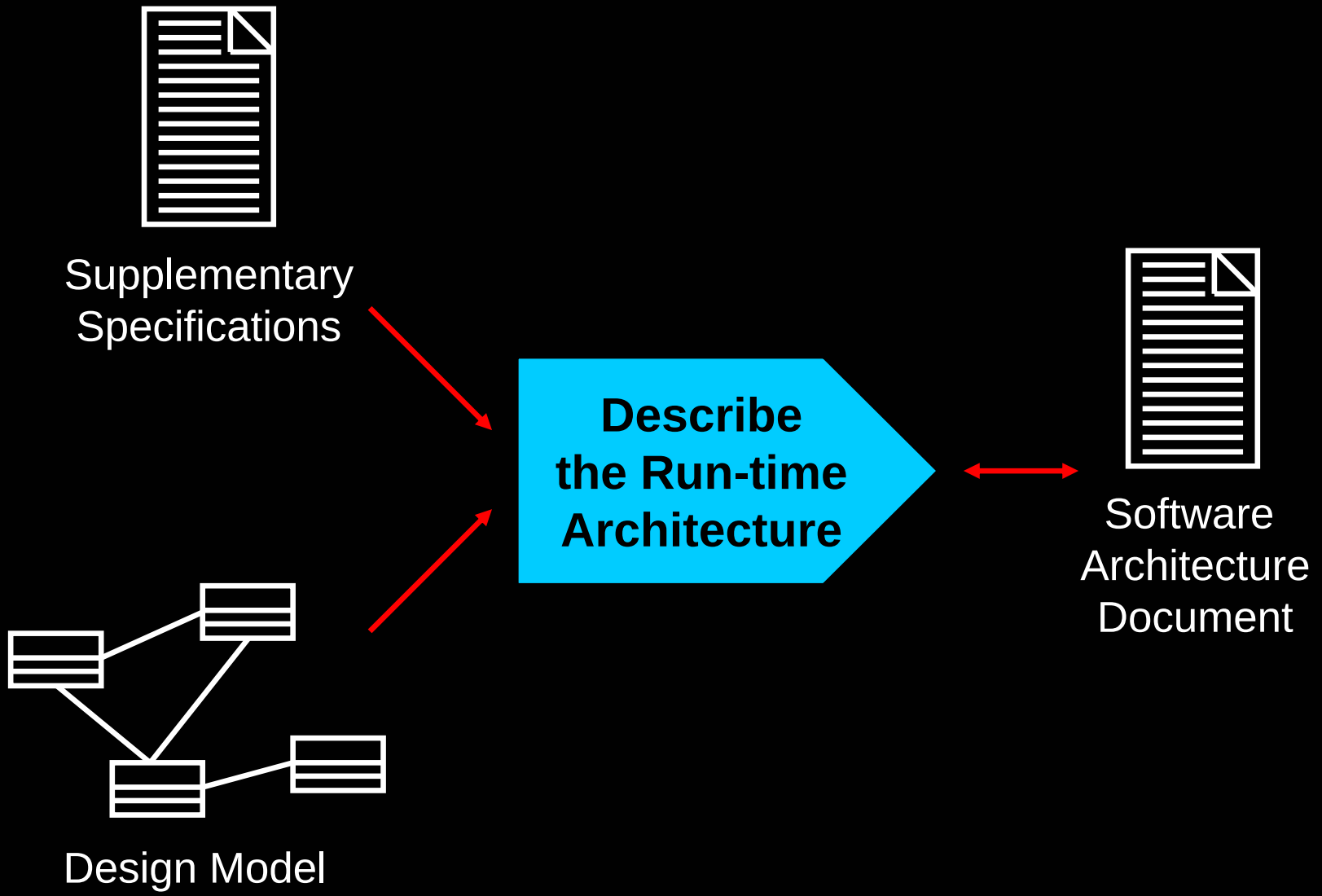
Objectives: Describe the Run-time Architecture

- 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

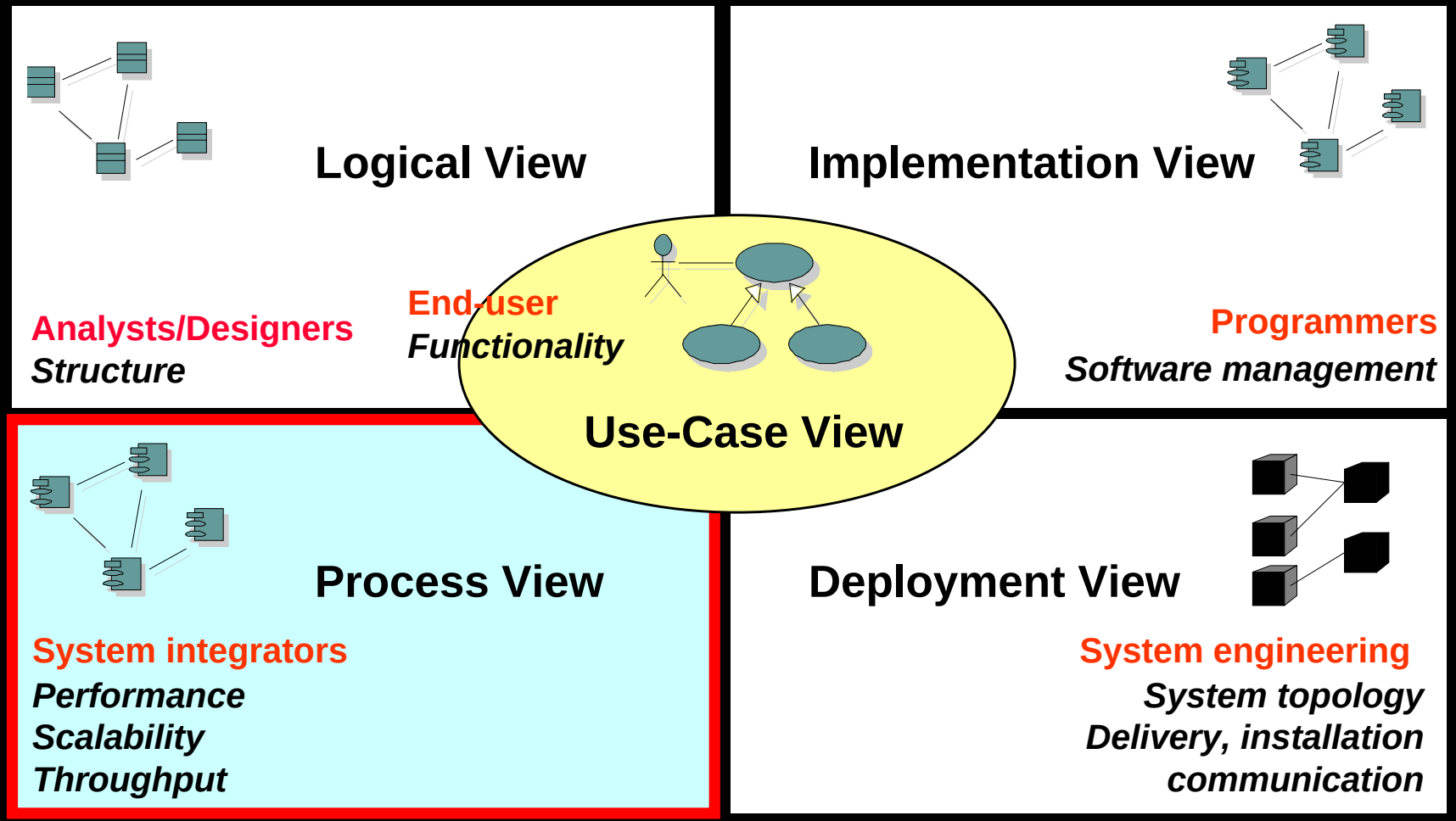
Describe the Run-time Architecture in Context



Describe the Run-time Architecture Overview



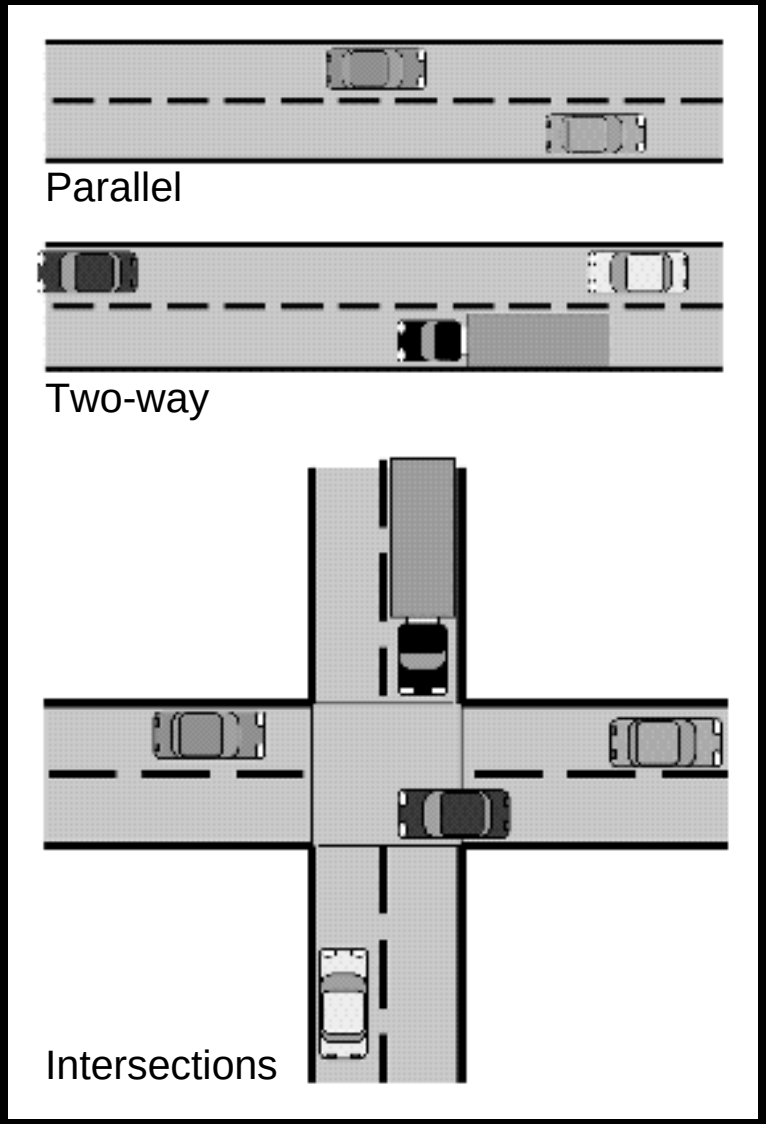
Key Concepts: The Process View



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

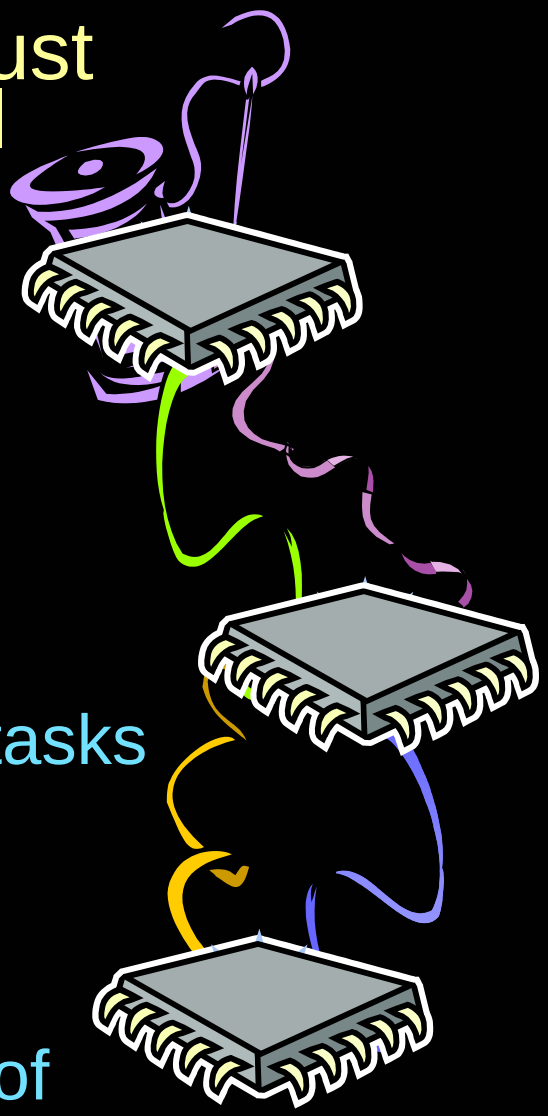


Why Are We Interested in Concurrency?

- 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 on a single CPU by interleaving the execution of different tasks
  - Application-based solutions
    - the application software takes responsibility for switching between different branches of code at appropriate times

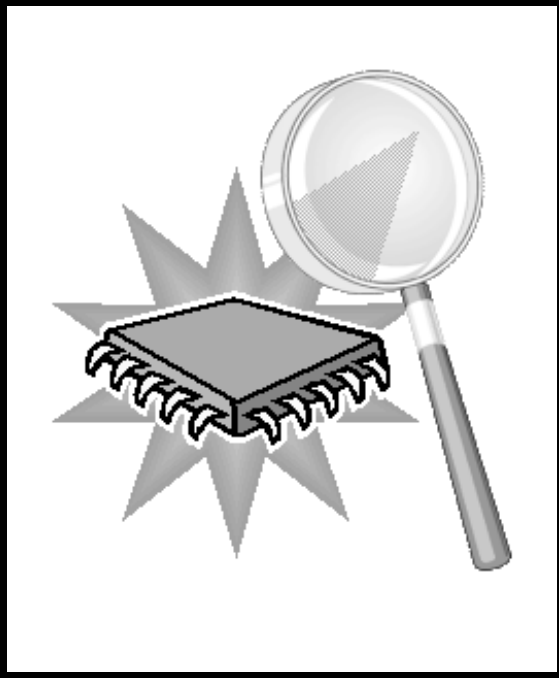


Describe the Run-time Architecture Steps

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

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

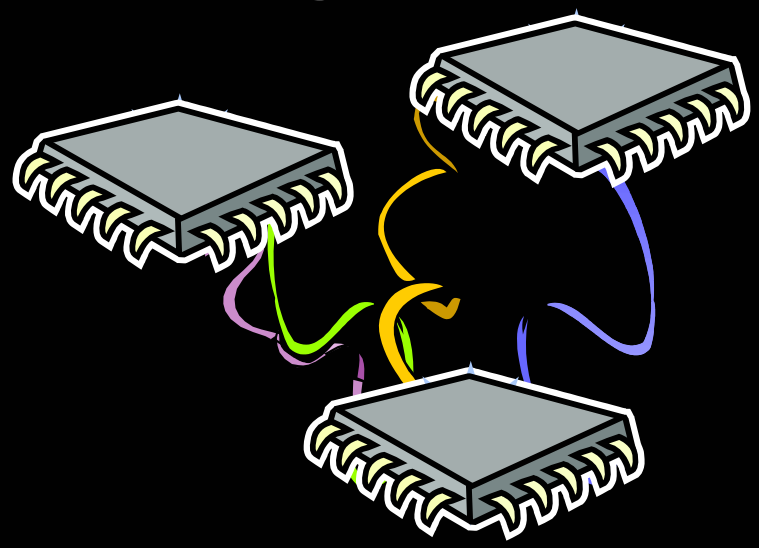
- Concurrency requirements are driven by:
  - The degree to which the system must be distributed.
  - The degree to which the system is event-driven.
  - The computation intensity of key algorithms.
  - The degree of parallel execution supported by the environment
- Concurrency requirements are ranked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

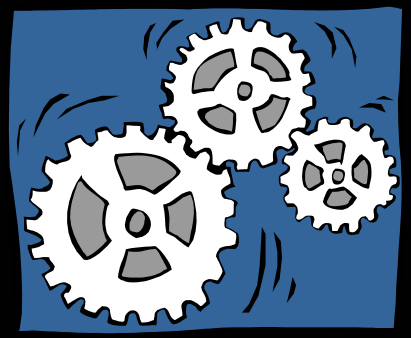
Describe the Run-time Architecture Steps

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

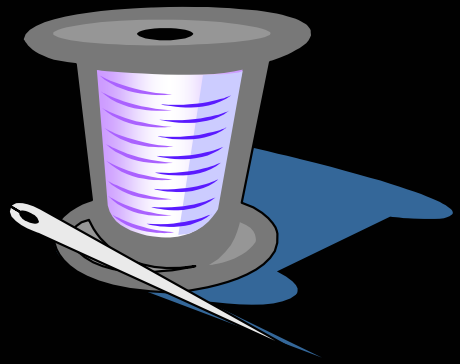


Key Concepts: Process and Thread

- 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

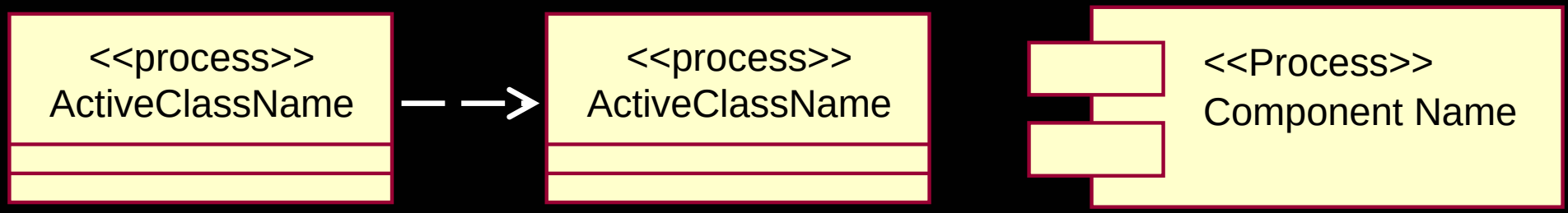


Identifying Processes and Threads

- 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 areas
    - Improvement of system availability
    - Support major subsystems

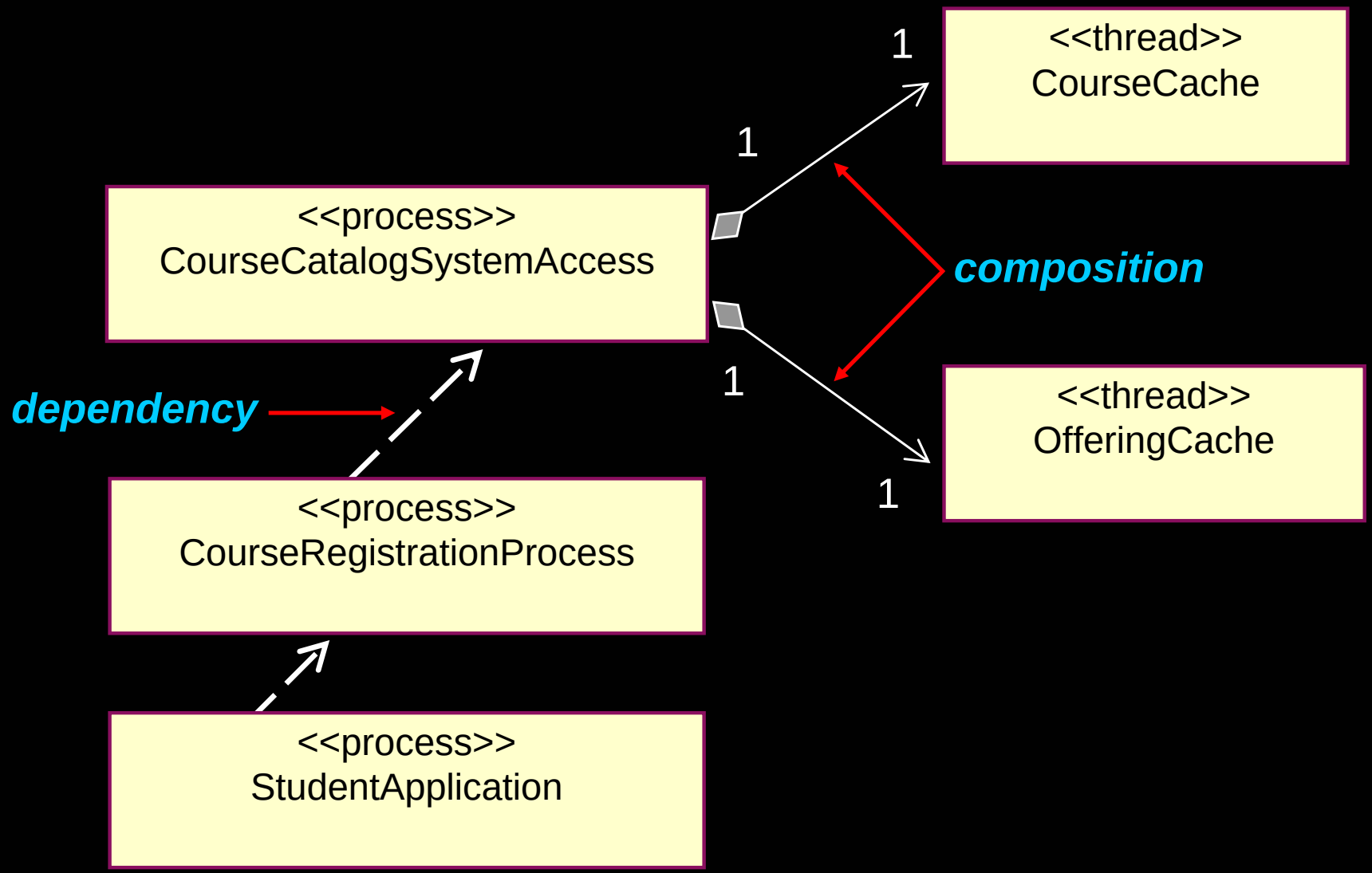
Modeling Processes

- 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

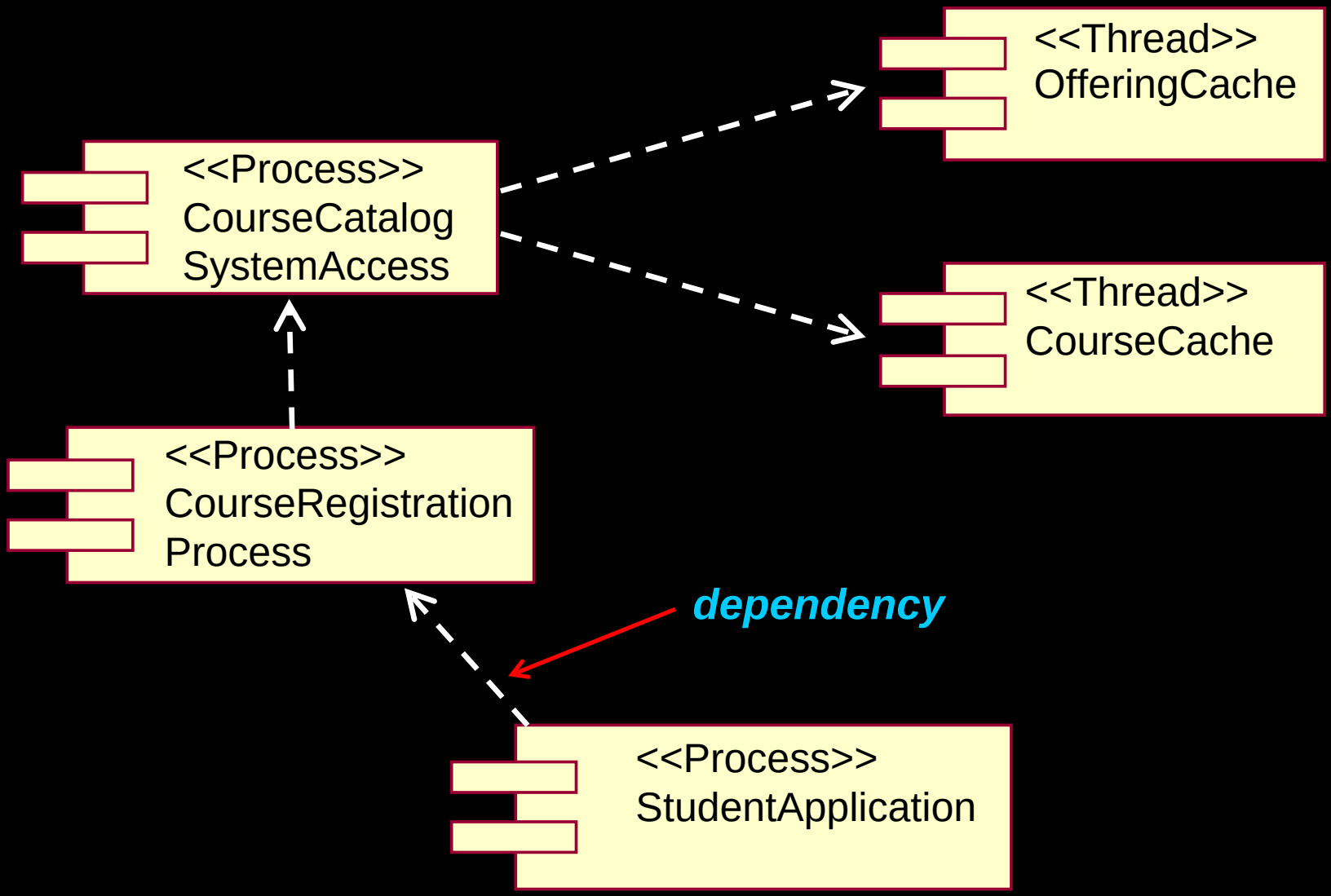


This course will model processes and threads using Class Diagrams.

Example: Modeling Processes: Class Diagram

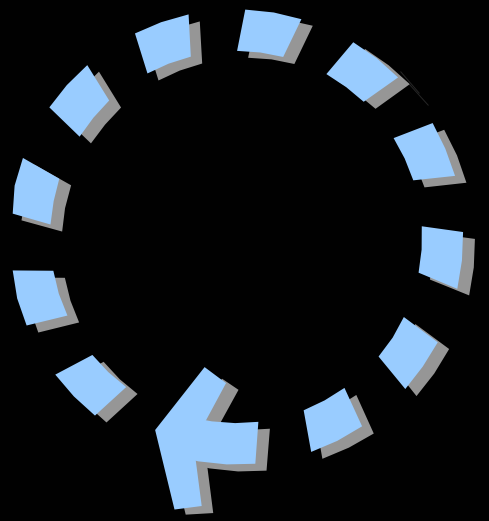


Example: Modeling Processes: Component Diagram



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

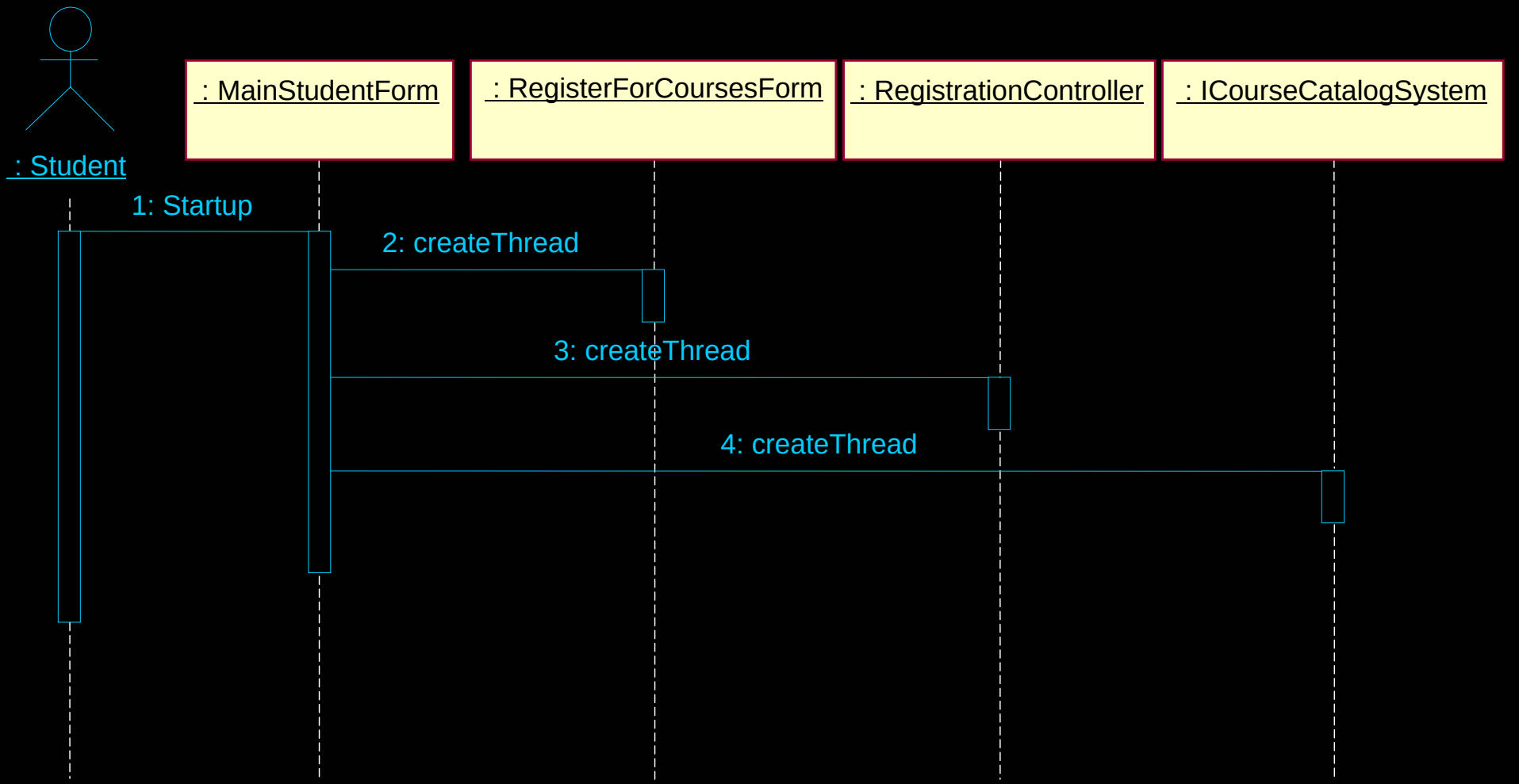


Creating and Destroying Processes and Threads

- 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

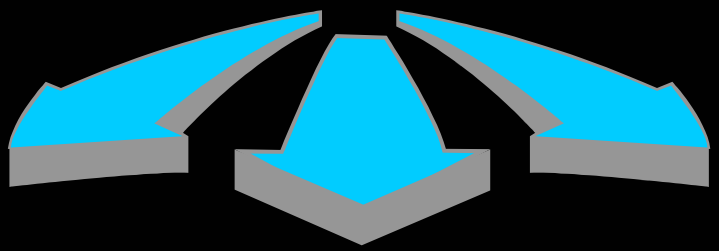
- Analyze concurrency requirements
- Identify processes and threads
- Identify process lifecycles
- ★ Map processes onto the implementation
- Distribute model elements among processes

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 resource availability

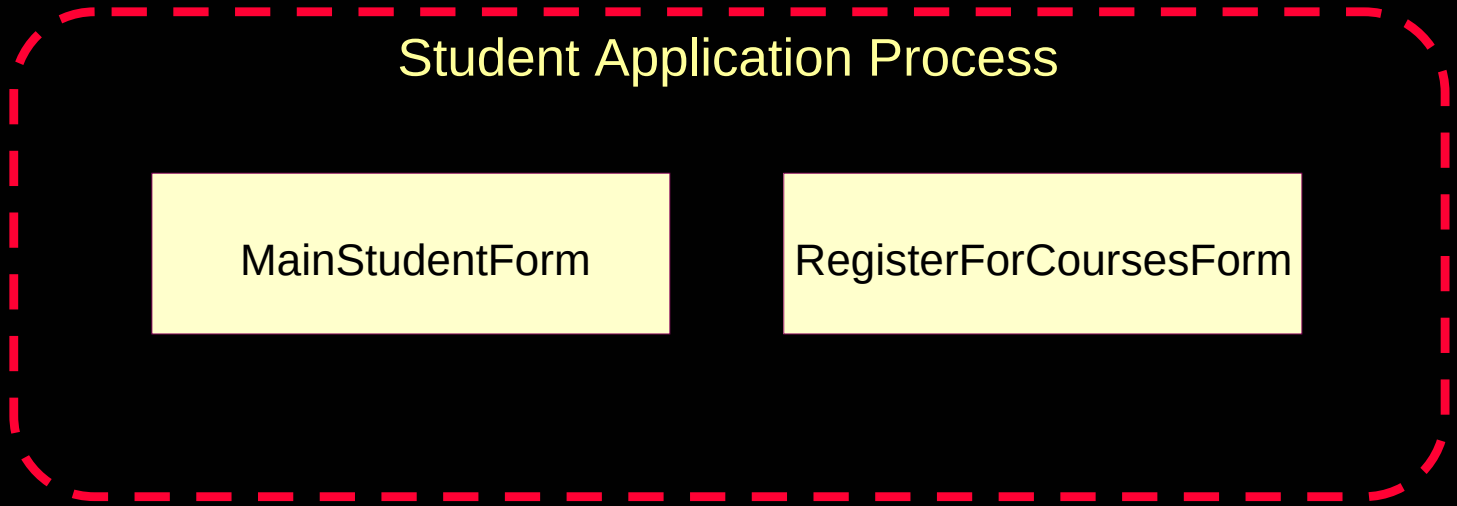
Describe the Run-time Architecture Steps

- 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

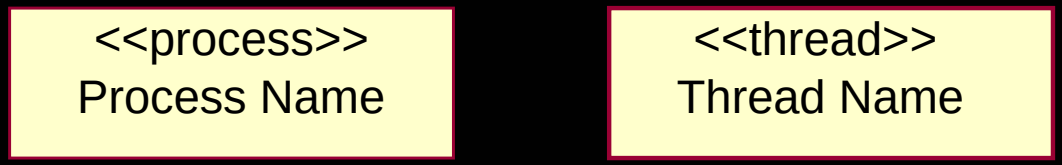
Design Elements-to-Processes Strategies

Two Strategies (used simultaneously)

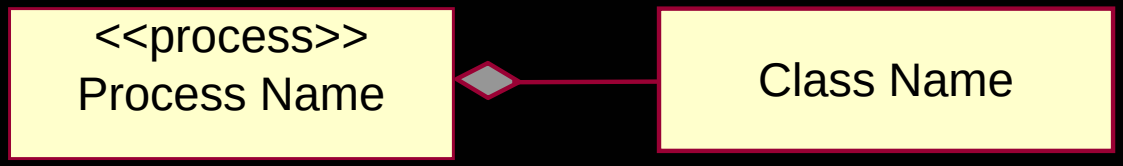
- Inside-Out
  - Group elements that closely cooperate and must execute in the same thread of control
  - Separate elements that do not interact
  - Repeat until you reach the minimum number of processes that still provide the required distribution and effective resource utilization
- Outside-In
  - 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

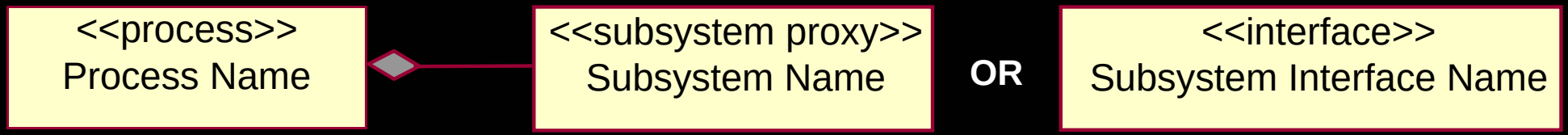
- Class diagrams
  - Active classes as processes



  - Composition relationships from processes to classes

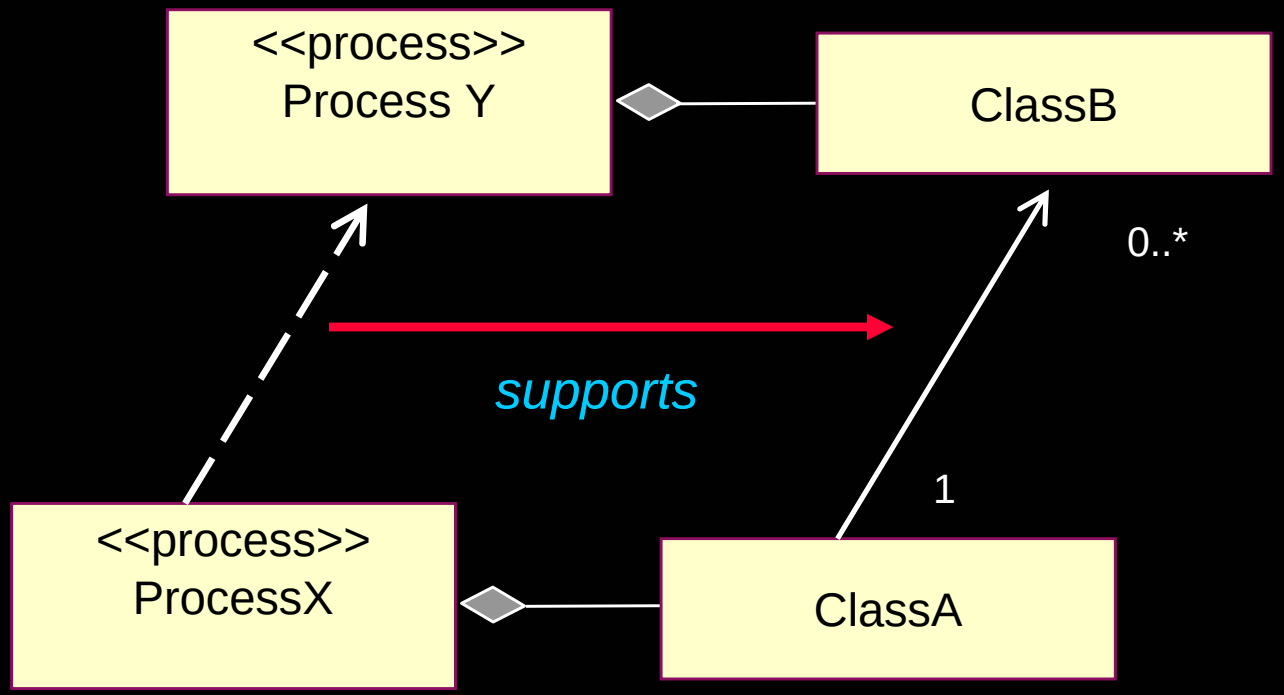


  - Composition relationships from processes to subsystems

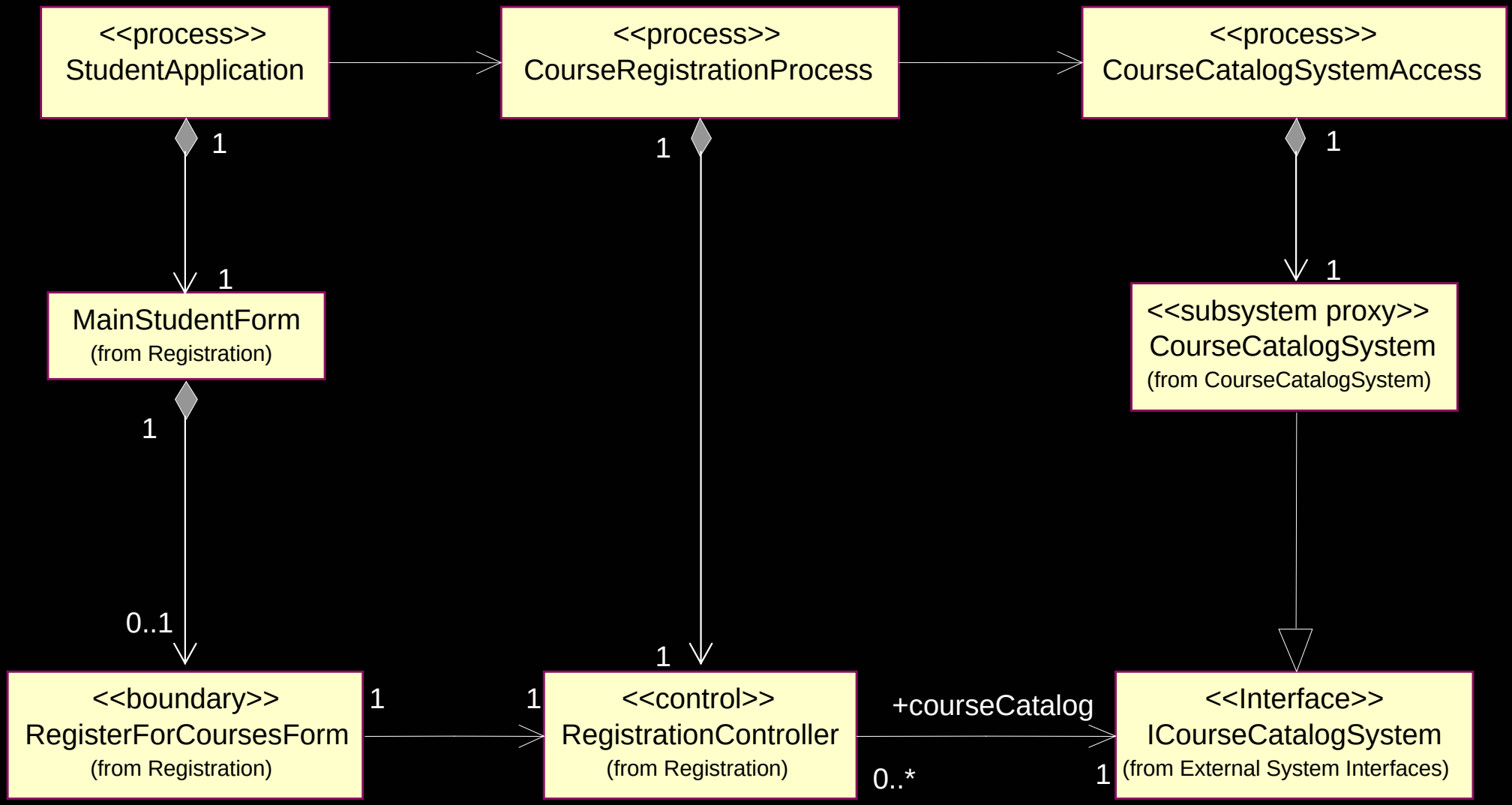


Process Relationships

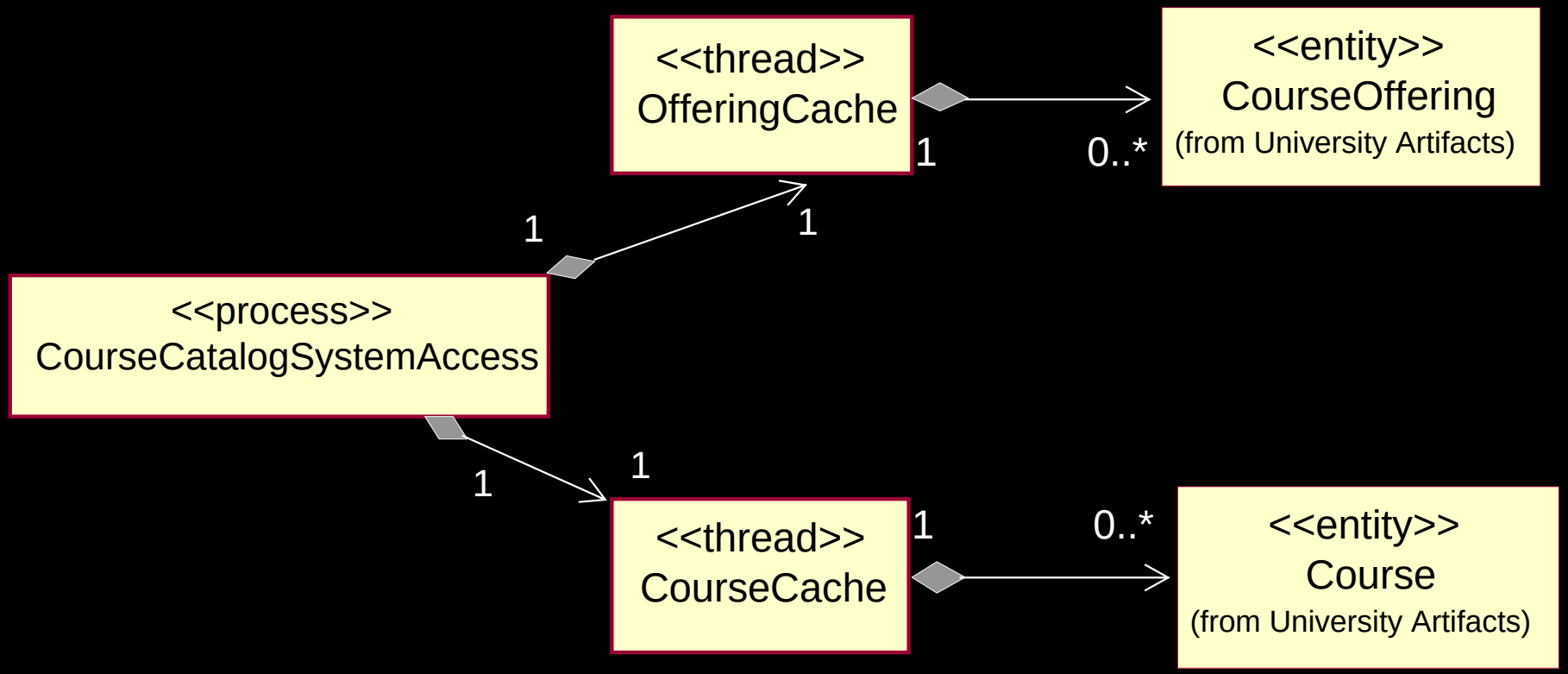
- Process relationships must support design element relationships



Example: Register for Course Processes



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?

Review: Describe the Run-time Architecture

- What is the purpose of the Describe the Run-time Architecture 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?

Exercise: Describe the Run-time Architecture

- Given the following:
  - Design elements (classes and subsystems) and their relationships
  - Processes
  - What classes and subsystems are mapped to what processes?

(continued)

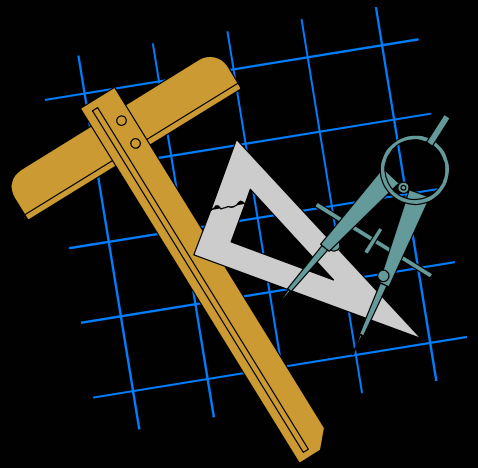
Exercise: Describe the Run-time Architecture (cont.)

- Identify the following:
  - Process relationships

(continued)

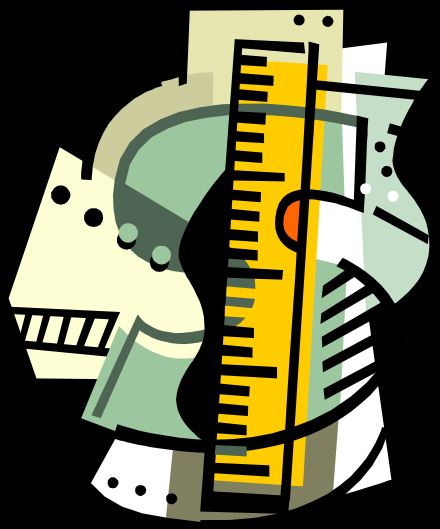
Exercise: Describe the Run-time Architecture

- 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: 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?



Payroll System