SOFT 437

Software Performance Analysis

Chapter 4: Software Execution Model

Software Execution Model

- Constructed early in the development process to ensure that the chosen software architecture can achieve the required performance objectives
- Captures essential performance characteristics of the software
- Provides a static analysis of the mean, best and worst-case response time
- Characterizes the resource requirements of the proposed software alone, in the absence of other workloads, multiple users or delays due to contention for resources

Software Execution Model (con't)

- Software execution models are generally sufficient for identifying serious performance problems at the architectural and early design phases
- We can refine software execution model in the critical areas

The absence of problems in the software model does not mean that there are none

The Execution Graphs

- Execution graphs are one type of software execution model
  - Visual representation that helps to communicate execution behavior
- An execution graph is constructed for each performance scenario
- The graphs consist of nodes and arcs
  - Nodes represent processing steps – a collection of **operation invocations** and program statements that perform a **function** in the software system
  - Arcs represent the order of execution

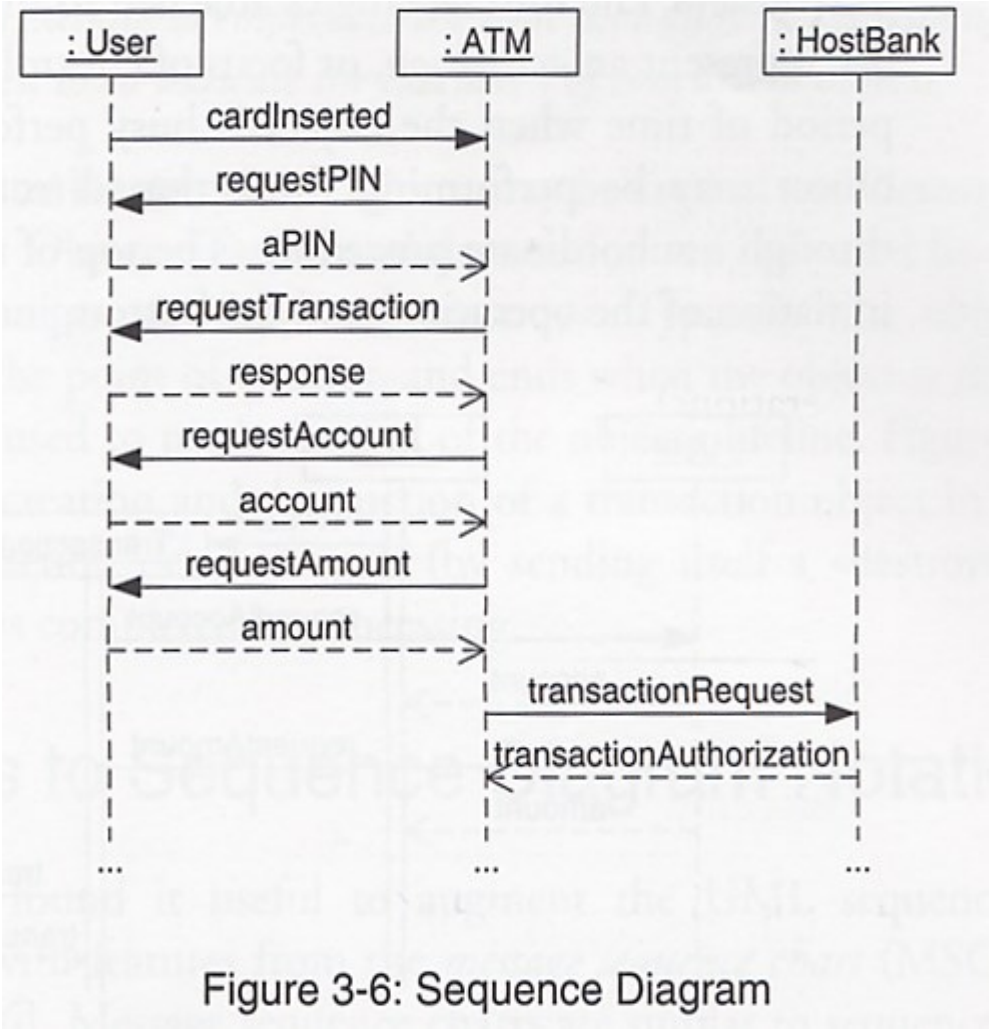


Figure 3-6: Sequence Diagram

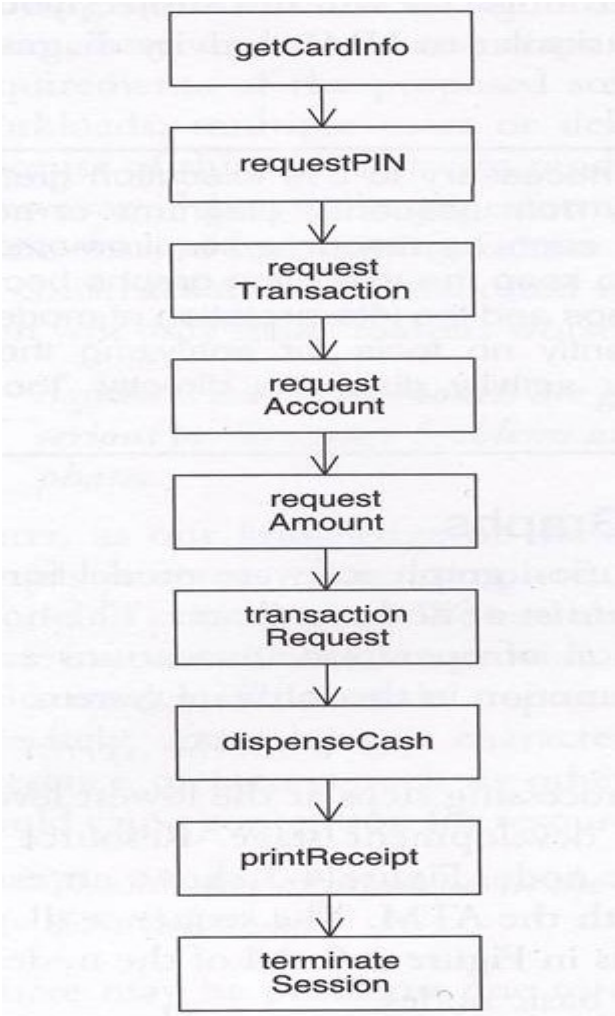


Figure 4-1: Sample Execution Graph

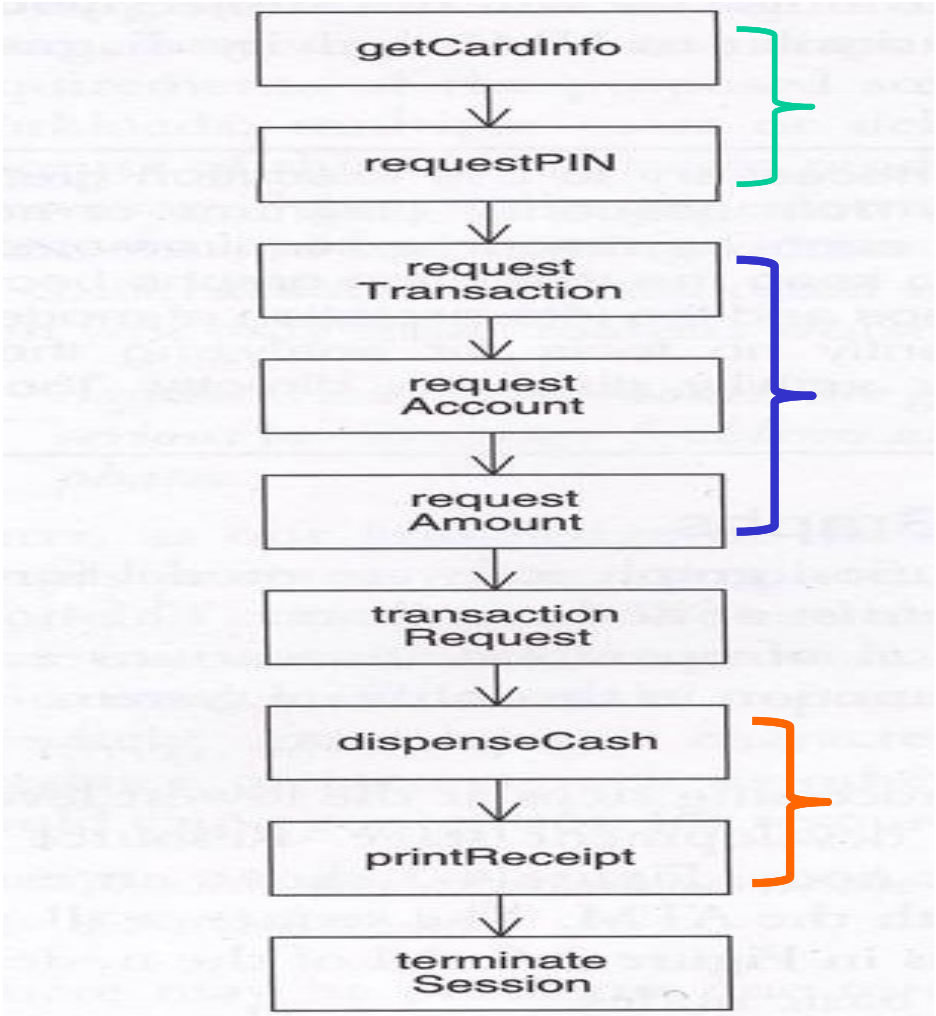


Figure 4-1: Sample Execution Graph

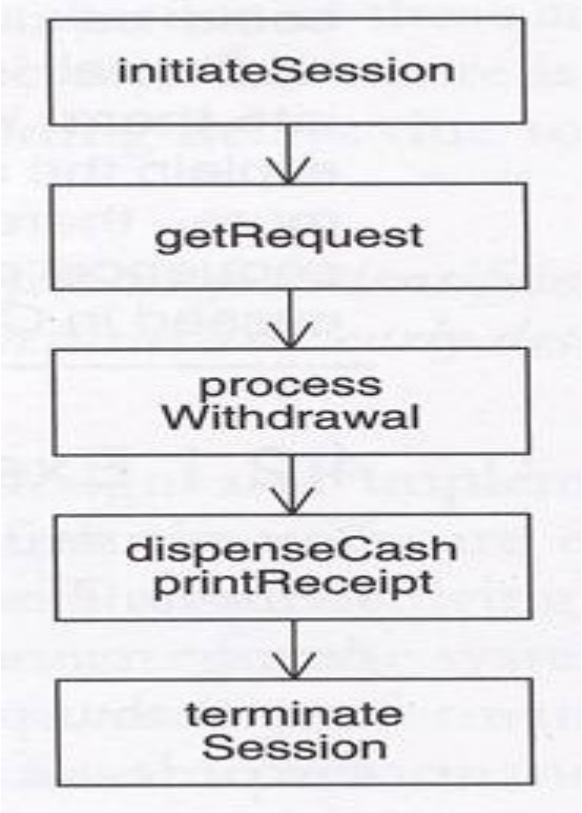


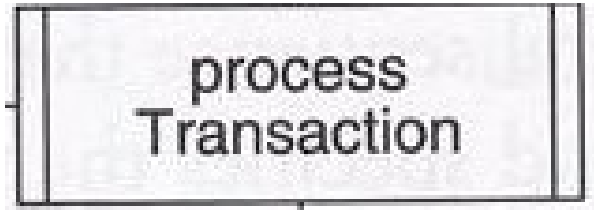
Figure 4-2: **Abbreviated Graph**

Basic Nodes

- ***Basic nodes*** (representation symbols) represent processing steps at the lowest level of detail that is appropriate for the current development stage
- Software execution models elaborate details of interest to performance
- The ***simple-model*** principle recommends that details that are not pertinent to performance should be excluded

Expanded Nodes

- Expanded nodes represent processing steps elaborated in another subgraph
- Expanded nodes show additional processing details that are identified as the design evolves



Expanded Nodes

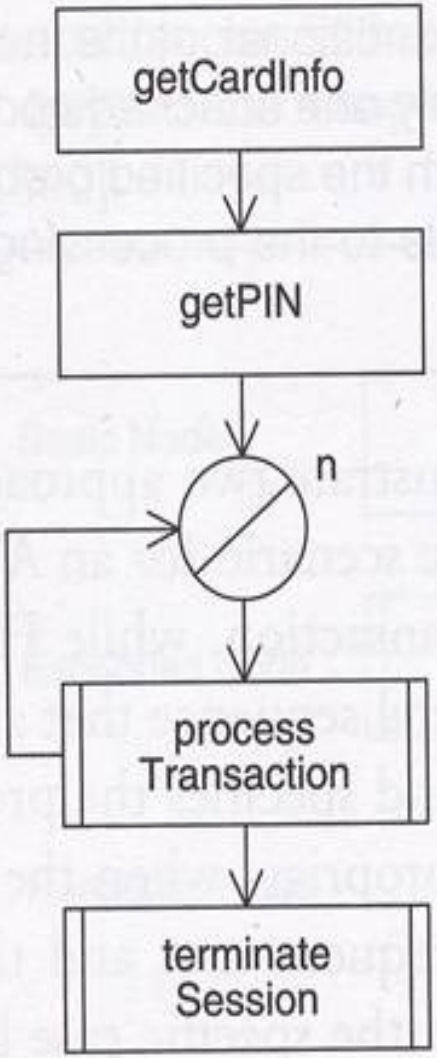


Figure 4-3: Execution Graph for General ATM Scenario

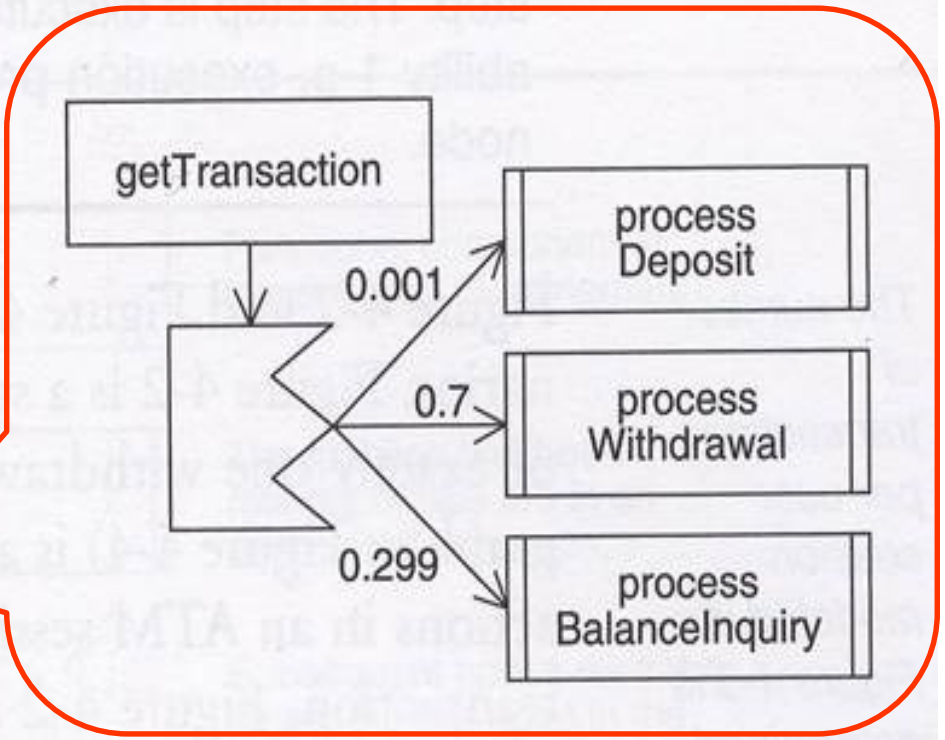
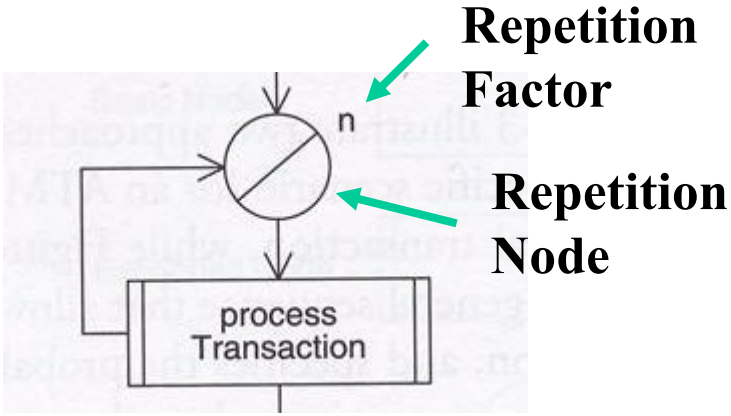


Figure 4-4: Subgraph for processTransaction Expanded Node

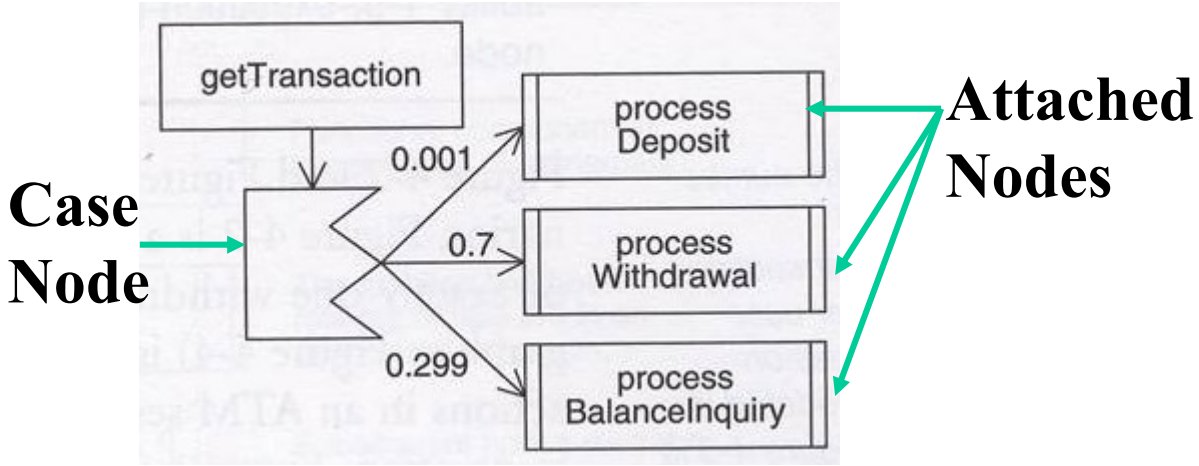
Repetition Nodes

- ***Repetition nodes*** represent one or more nodes that are repeated
- ***Repetition factor*** associated with the node that specifies the number of times the processing steps repeat
- An arc connects the last node repeated with the repetition node



Case Nodes

- ***Case nodes*** represent conditional execution of processing steps
- ***Attached nodes*** represent the steps that may be executed
- A case node has more than one attached
- Each attached node has an execution probability



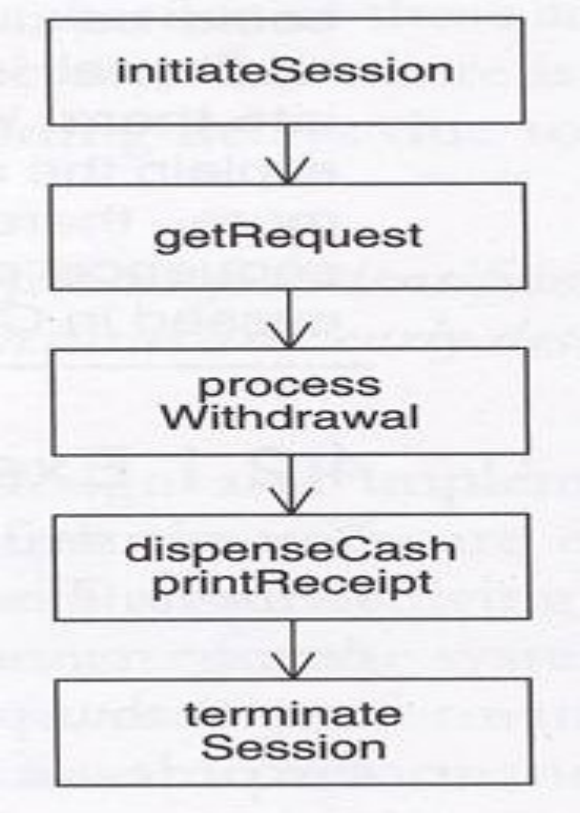


Figure 4-2: **Abbreviated Graph**

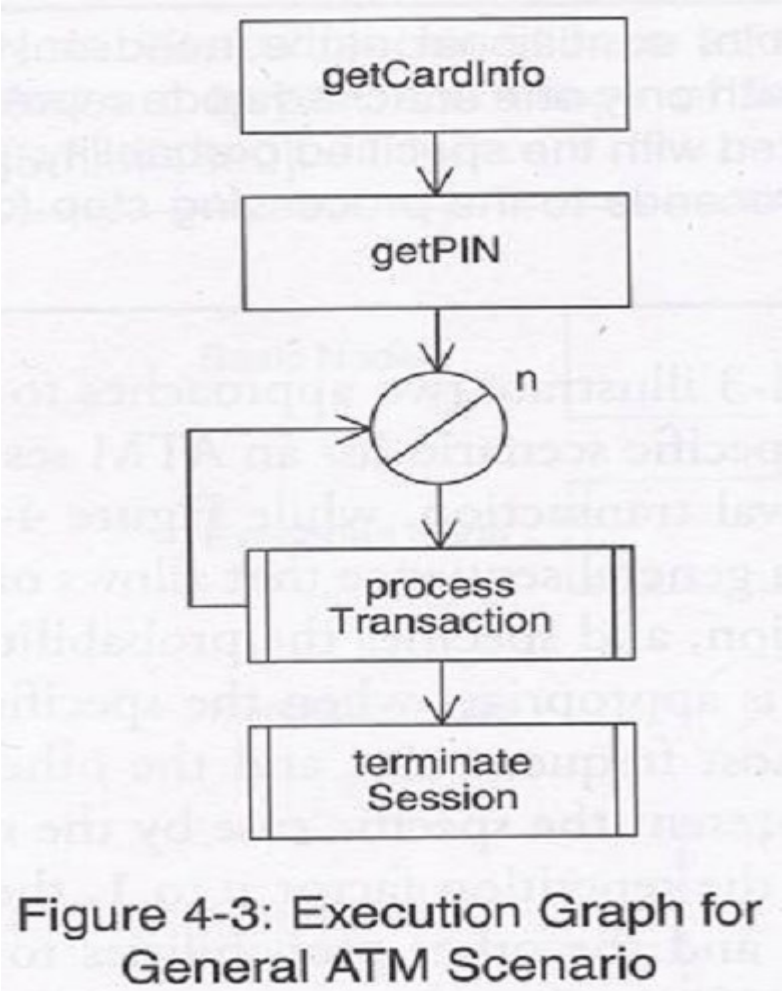


Figure 4-3: Execution Graph for General ATM Scenario

Pardo Node

- ***A Pardo*** (as Parallel do) ***node*** represents parallel execution within a scenario

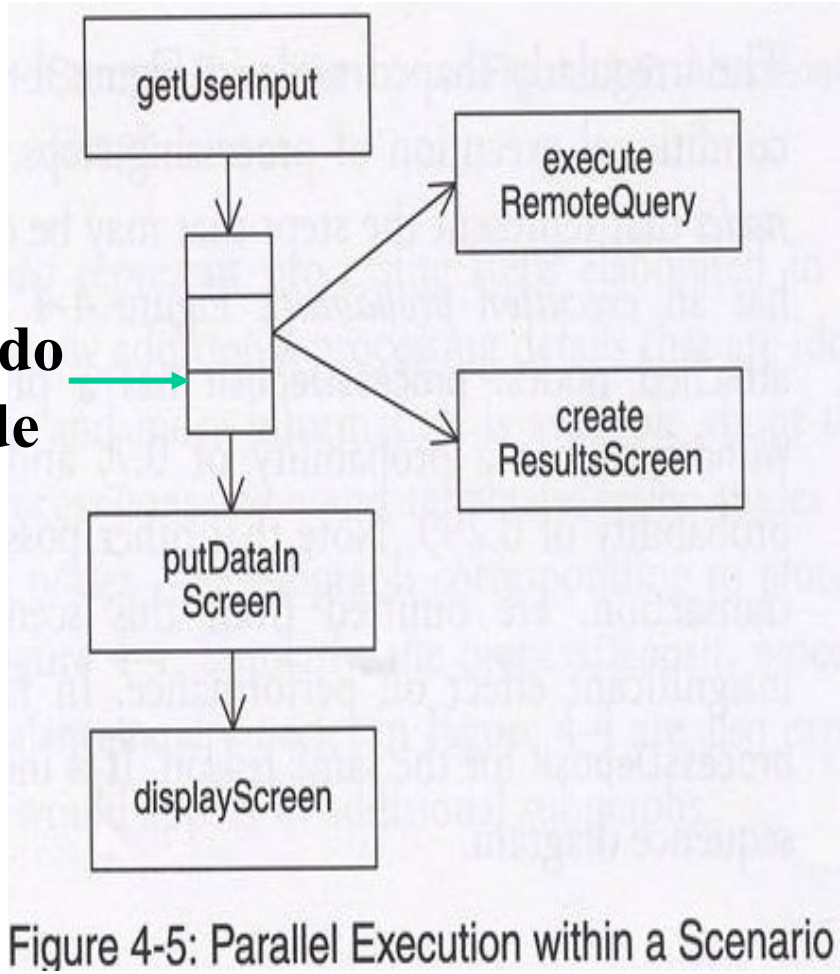


Figure 4-5: Parallel Execution within a Scenario

Basic Execution Graph Notation

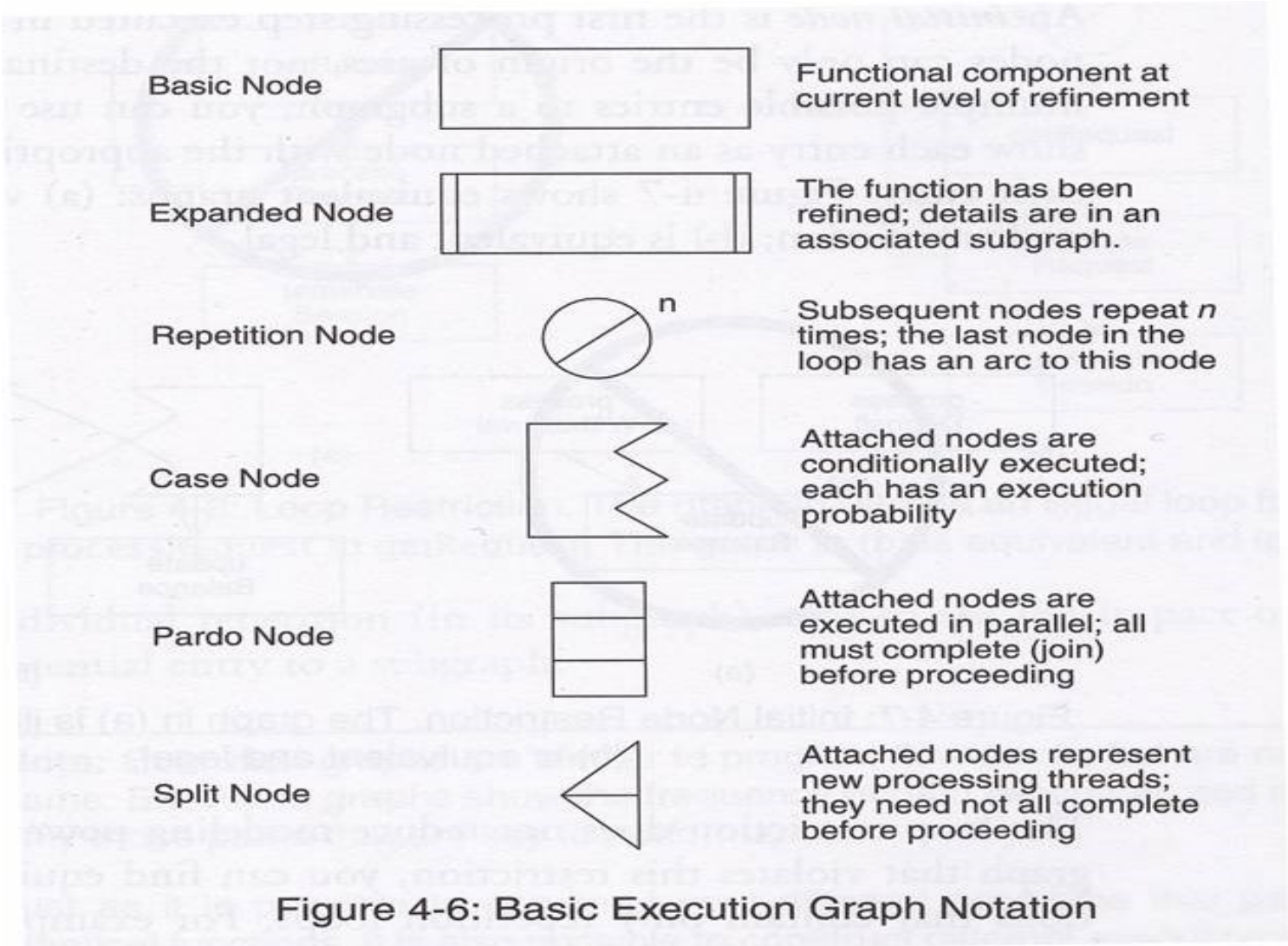
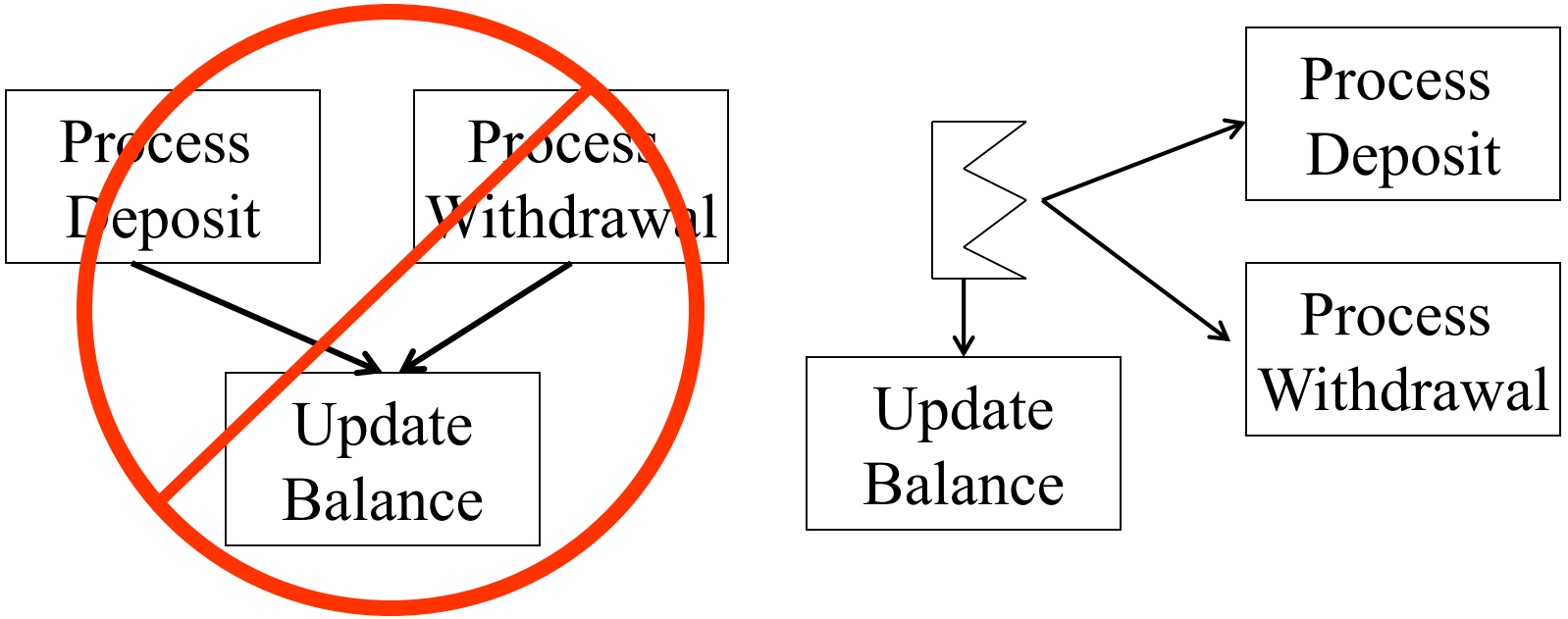


Figure 4-6: Basic Execution Graph Notation

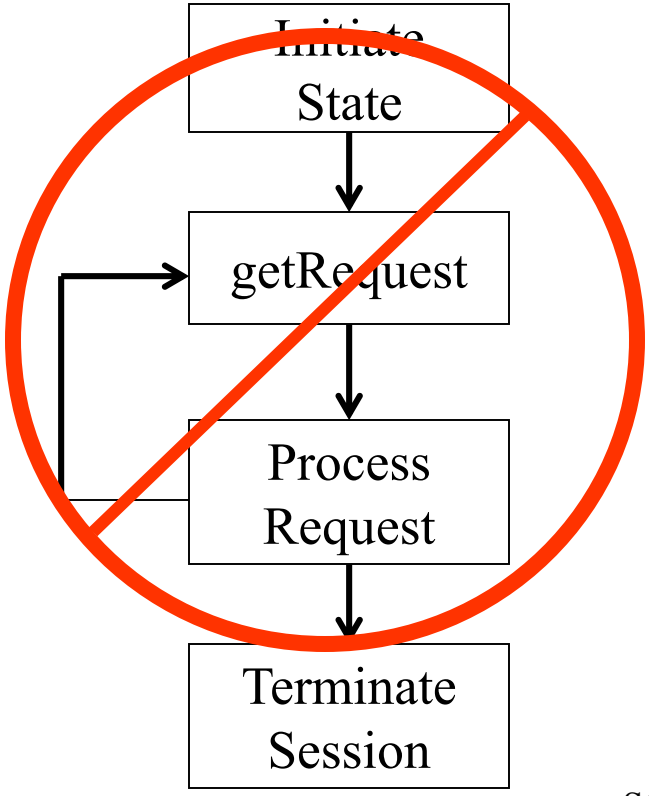
Graph Restrictions

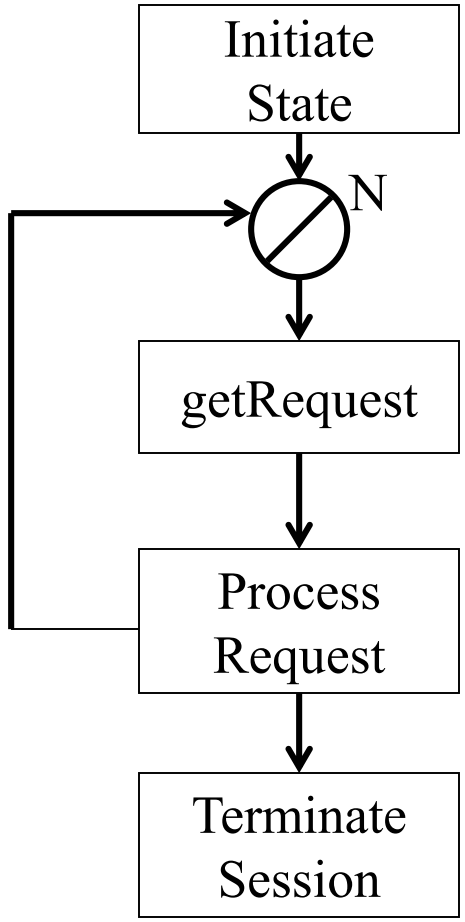
- Initial node restriction: graphs and subgraphs can have only one initial node

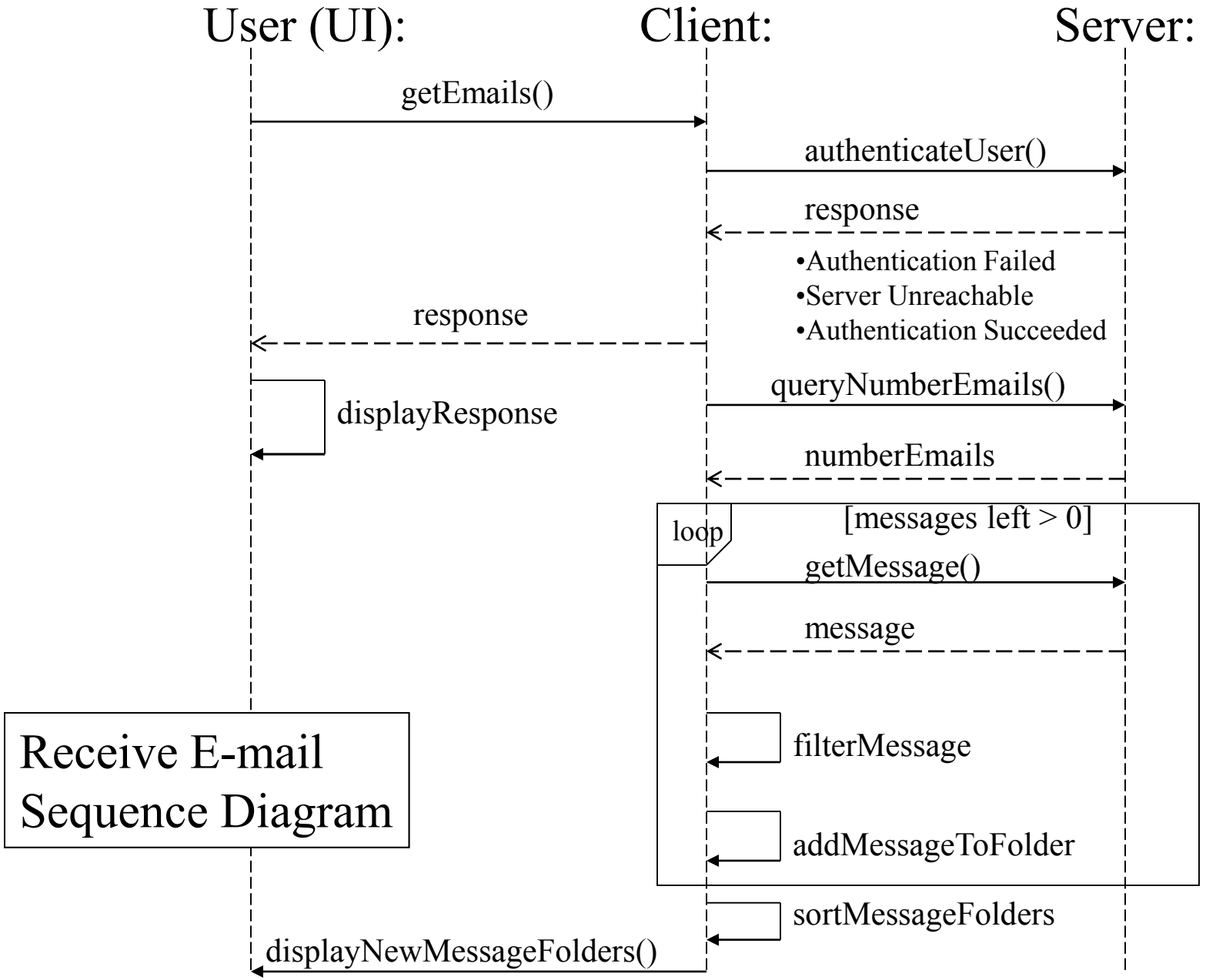


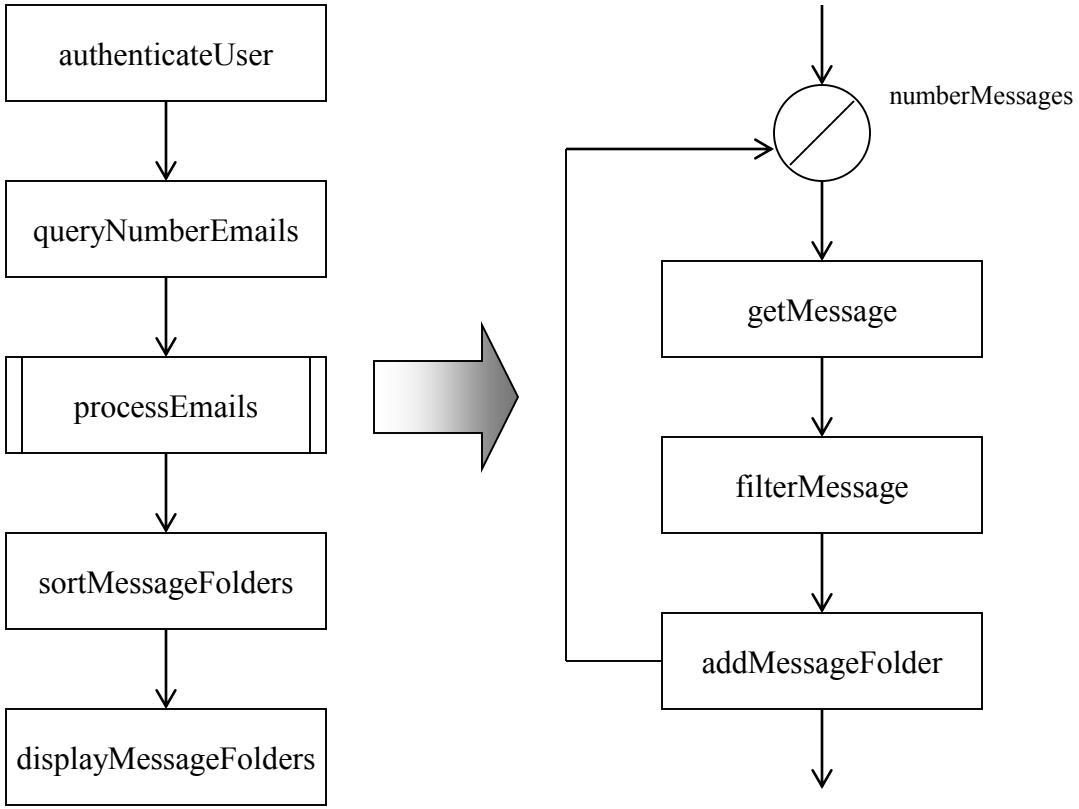
Graph Restrictions

- Loop restriction: all loops in the graph must be repetition loops






User (UI):
Client:
Server:
getEmails()
authenticateUser()
response
•Authentication Failed
•Server Unreachable
•Authentication Succeeded
response
displayResponse
queryNumberEmails()
numberEmails
loop
[messages left > 0]
getMessage()
message
filterMessage
addMessageToFolder
sortMessageFolders
displayNewMessageFolders()
Receive E-mail Sequence Diagram



Receive E-mail Software Execution Model

Software Execution Model Analysis

- Primary purposes of software execution model analysis are
  - Make a quick check of the best-case response time in order to ensure the architecture and design will lead to satisfactory performance
  - Assess the performance impact of alternatives
  - Identify critical parts of the system for performance management
  - Derive parameters for the system execution model
- The algorithms are formulated for evaluating graphs

Basic Solution Algorithms

- The algorithms are ‘easy’ to understand
  - Examine graphs and identify a basic structure
  - Compute the time of a basic structure and reduce the basic structure to a ‘computed node’
  - Continue until only one node left
- Basic structures are
  - Sequences
  - Loops
  - Cases

Graph Reduction for Sequential Structures

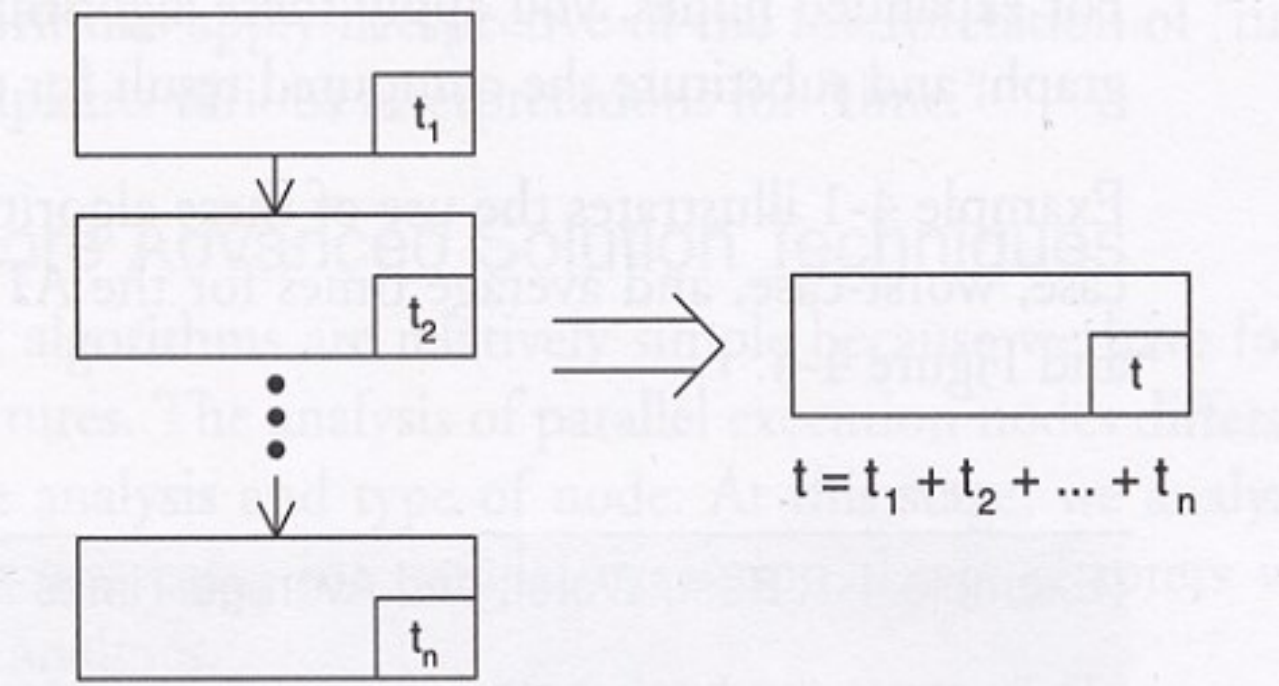


Figure 4-9: Graph Reduction for Sequential Structures

Graph Reduction for Loop Structures

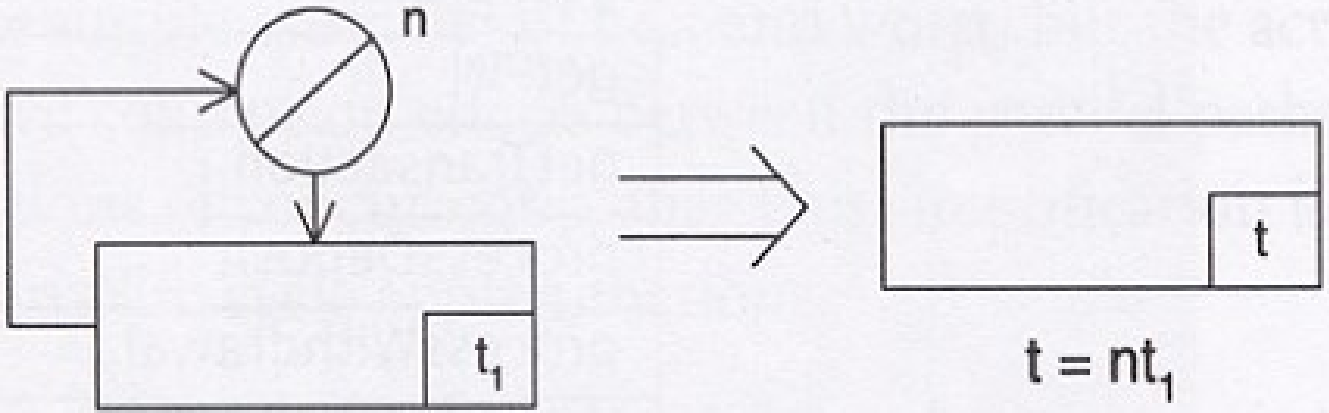


Figure 4-10: Graph Reduction for Loop Structures

Graph Reduction for Case Nodes

- The computation for case nodes differs for shortest path, longest path, and average analyses
  - Shortest path: the time for the case node is the minimum of the times for the conditionally executed nodes
  - Longest path: the time for the case node is the maximum of the times for the conditionally executed nodes
  - For the average analysis: the time is multiplying each node's time by its execution probability

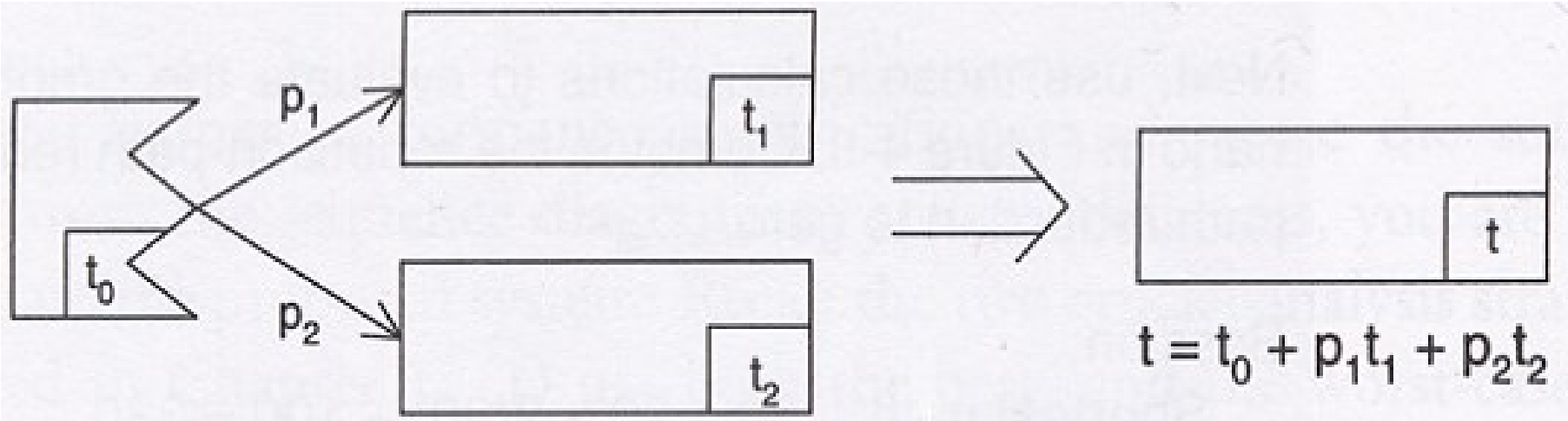


Figure 4-11: Graph Reduction for Case Nodes

Example: ATM Scenario

Example 4-1: Best, Worst and Average Times for ATM Scenario
To illustrate the basic path reductions, consider the ATM scenario in Figure 4-3 and the subgraph for processTransaction in Figure 4-4. Assume the node "times" in the following table.

Node	Time
getCardInfo	50
getPIN	20
getTransaction	30
processDeposit	500
processWithdrawal	200
processBalanceInquiry	50
terminateSession	100

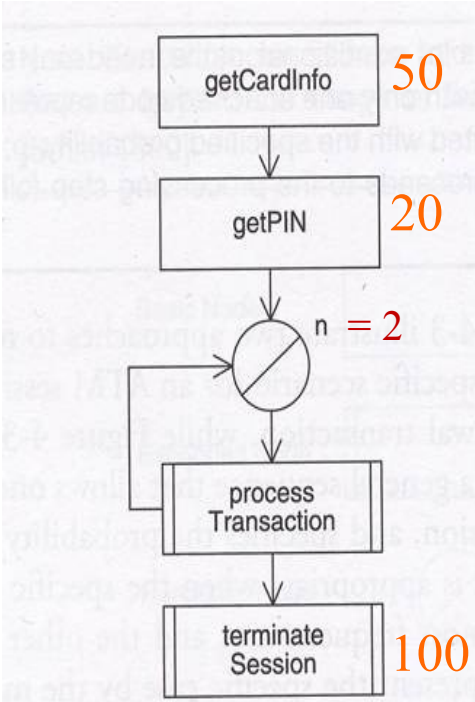


Figure 4-3: Execution Graph for General ATM Scenario

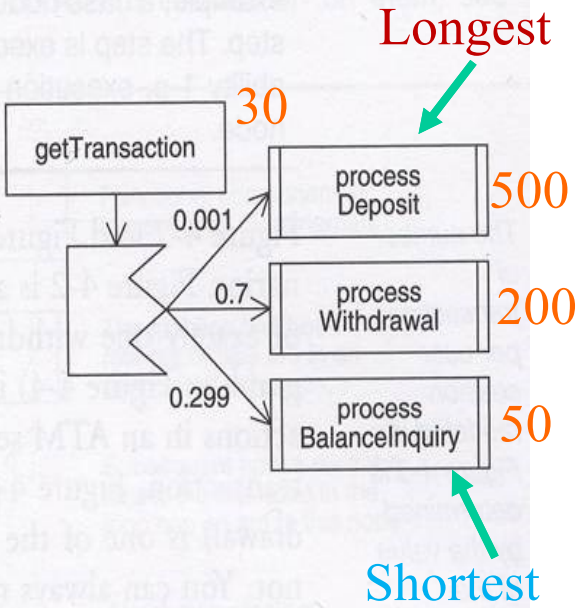


Figure 4-4: Subgraph for processTransaction Expanded Node

Analysis Procedures

- Use both the best- and the worst-case estimates of resource requirements for each basic node
- Begin with a simplistic analysis of the best case and introduce more sophisticated analyses of realistic cases as more detailed information becomes available

Software Resource Requirements

- Each basic node has specified SW resource requirements A_j for each service unit j, e.g.

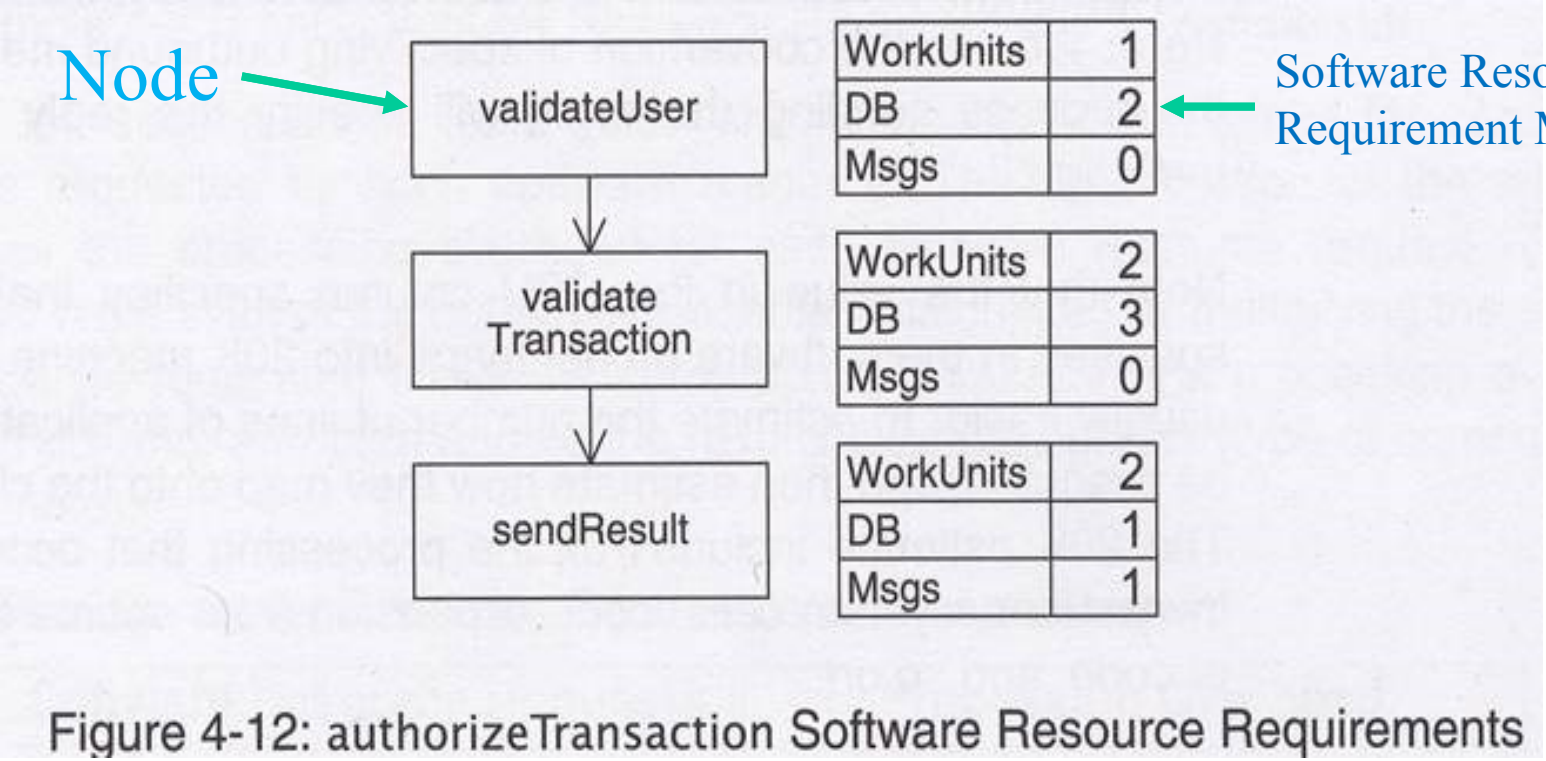


Figure 4-12: authorizeTransaction Software Resource Requirements

Processing Overhead Matrix

- A chart of the computer resource requirements for each of the software resource requests

Table 4-1: Processing Overhead

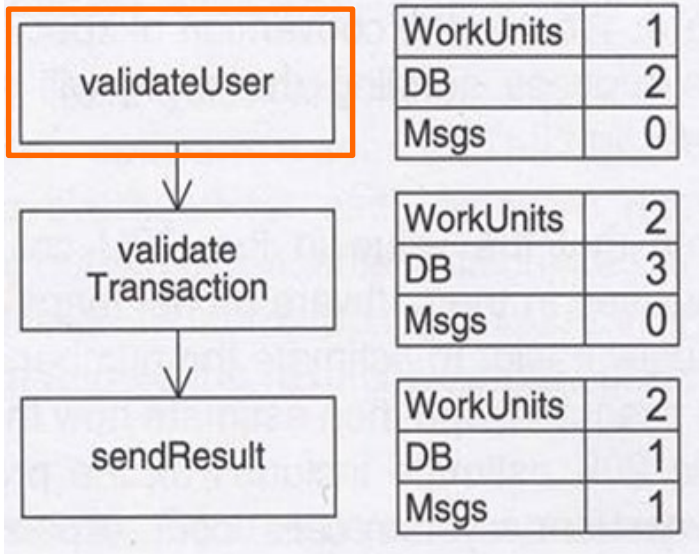
Device	CPU	Disk	Network
Quantity	1	1	1
Service Unit	KInstr.	Phys. I/O	Msgs.
WorkUnit	20	0	0
DB	500	2	0
Msgs	10	2	1
Service time	0.00001	0.02	0.01

Hardware Resources

Mapping between software resource requirements and computer device usage

Table 4-1: Processing Overhead

Device	CPU	Disk	Network
Quantity	1	1	1
Service Unit	KInstr.	Phys. I/O	Msgs.
WorkUnit	20	0	0
DB	500	2	0
Msgs	10	2	1
Service time	0.00001	0.02	0.01



Software Resource Requirements

		KInstr.	Phys. I/O		Msgs.
WorkUnits	1	20	0		0
DB	2	500	2		0
Msgs	0	10	2		1
validateUser		**1020**	**4**		**0**
		0.00001	**0.02**		**0.01**

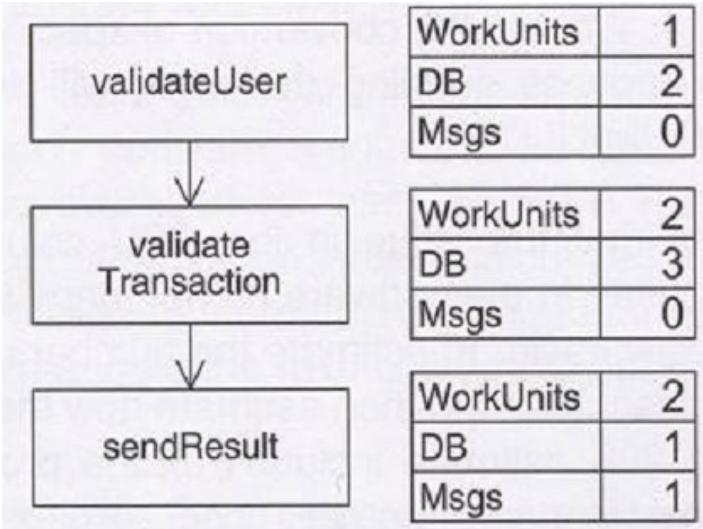
X

Computing the total execution time

- ***STEP 1:*** uses the processing overhead matrix to calculate the total computer resources required per software resource for each node in the graph

Total Computer Resource Requirements for sendResult

Software Resource Requests		Processing Overhead		
Name	Service Units	CPU Kinstr	Physical I/O	Network Messages
WorkUnit	2	20	0	0
DB	1	500	2	0
Msgs	1	10	2	1
Total: sendResult		550	4	1

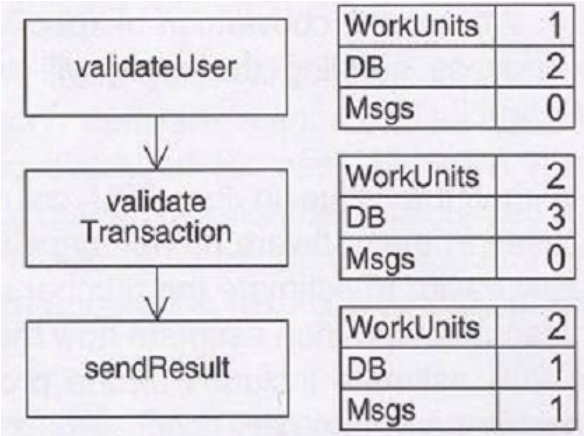


Computing the total execution time

- ***STEP 2:*** computes the total computer resource requirements for the graph

Processing Step	CPU Kinstr	Physical I/O	Network Messages
validateUser	1,020	4	0
validateTransaction	1,540	6	0
sendResult	550	4	1
Total: authorizeTransaction	3,110	14	1

Table 4-2: Total Computer Resource Requirements for authorizeTransaction



Computing the total execution time

- ***STEP 3:*** compute the best-case elapsed time

Table 4-1: Processing Overhead

Device	CPU	Disk	Network
Quantity	1	1	1
Service Unit	KInstr.	Phys. I/O	Msgs.
WorkUnit	20	0	0
DB	500	2	0
Msgs	10	2	1
Service time	0.00001	0.02	0.01

Processing Step	CPU Kinstr	Physical I/O	Network Messages
validateUser	1,020	4	0
validateTransaction	1,540	6	0
sendResult	550	4	1
Total: authorizeTransaction	3,110	14	1

Types of Software Resource

Software Resource Types	Description
CPU usage	specified in work units, estimated number of instructions executed, or CPU time (when measurements of similar systems are available)
SQL	specified as the number and type of SQL statements (read, write, update, etc.) executed when the software accesses a database
File I/O	specified as the number of logical or physical I/Os
Messages	specified as the number or size in Kbytes of messages sent via a local LAN or other type of network, and the number or size of those sent via an external network such as a WAN, Internet connection, and so on

Types of Software Resource (con't)

Software Resource Types	Description
Logging to files or data-bases	specified as the number of log events that execute
Calls to middleware functions	specified as the number and type of call (e.g., con-nectionOpen, queueGet, requestSend, and so on) when applicable
Calls to software in a different process, thread, or processor	specified as the number, type, and target of the call. (Note that some software interactions may be mod-eled explicitly when the interacting software is also under study, as in the examples in Chapter 5.)
Delay for remote processing	specified as the estimated elapsed time for each request

Software Resource Estimation

- One of the most difficult resources to estimate is CPU usage
  - We use work units that focus on the relative amount of work performed in a processing step
- Early in development, models typically will use two to five types of software resource specifications
- Later, you may include more software resource types, such as synchronization and lock requests

Another Example of Processing Overhead Matrix

Table 7-2: Sample Processing Overhead Matrix

Devices	CPU	Disk	Delay	GlNet
Quantity	6	3	1	1
Service units	Sec.	PhysI/O	Sec.	Msgs
WorkUnits	0.01			
I/O	0.0005	1		
Msgs	0.0005	1		1
Delay			1	
Service time	1	0.003	1	0.05

Case Study: ICAD (Interactive CAD)

- Engineers will use the application to construct and view drawings that model structures, such as aircraft wings
- The system also allows users to store a model in a database, and interactively assess the design's correctness, feasibility, and suitability
- The model is stored in a relational data, and several versions of the model may exist within the database
- An ICAD drawing consists of nodes and elements (e.g., beans, triangles, or plates)

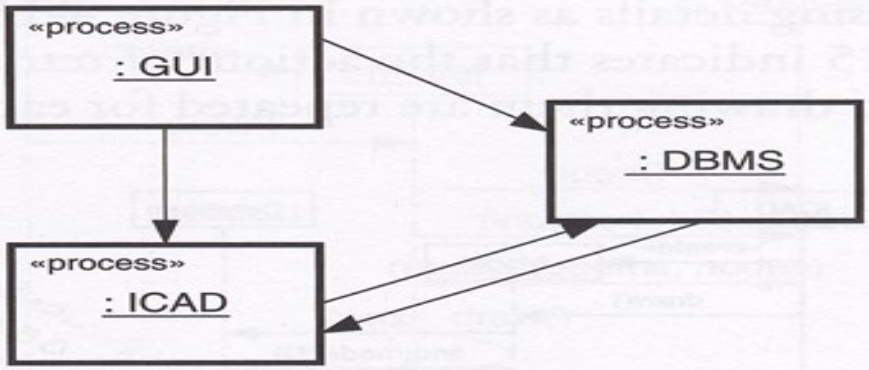


Figure 4-16: ICAD Processes

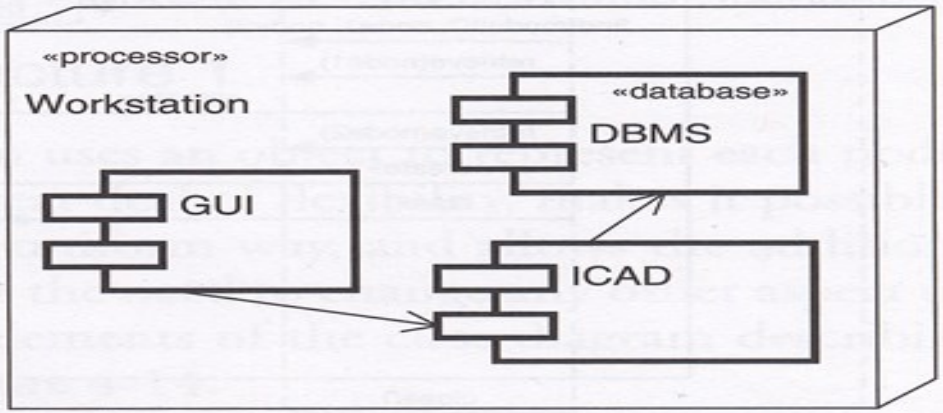
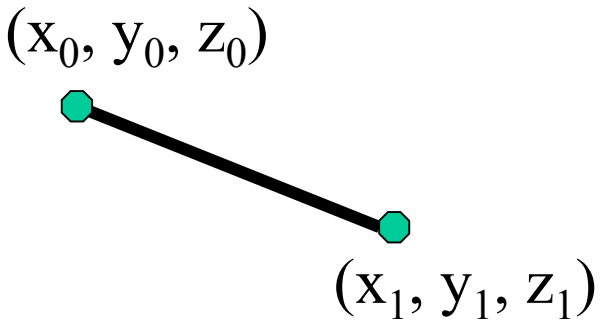
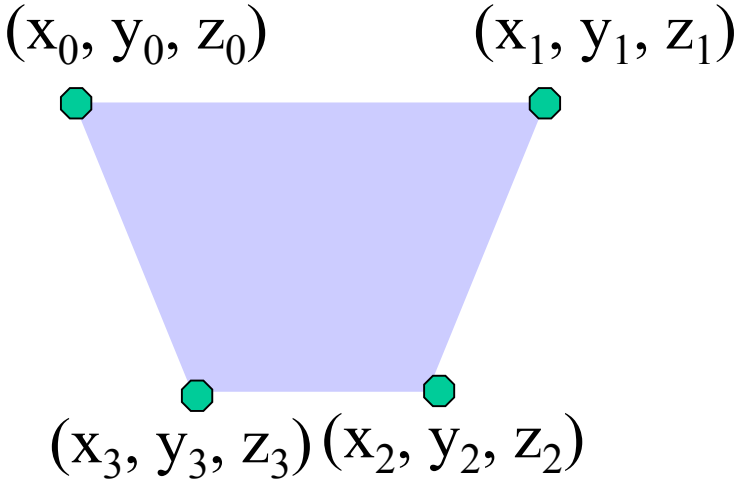
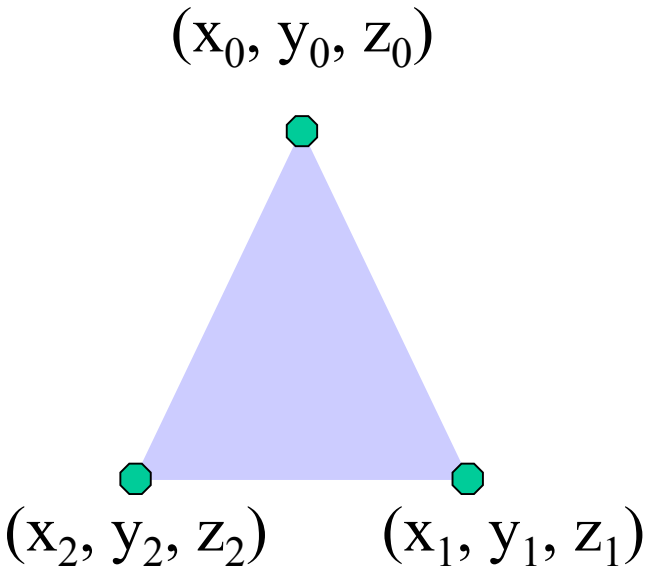


Figure 4-17: ICAD Deployment Diagram


(x_0, y_0, z_0)
(x_1, y_1, z_1)


(x_0, y_0, z_0)
(x_1, y_1, z_1)
(x_3, y_3, z_3)
(x_2, y_2, z_2)


(x_0, y_0, z_0)
(x_2, y_2, z_2)
(x_1, y_1, z_1)

Use Cases

- Use Case: **Draw** (draw a model), **Solve** (solve a model)
- Scenario: **DrawMod** (Draw models)
- A typical model contains only ***nodes*** and ***beams*** and consists of ***2,000 beams***.
- Performance goal is to draw (show on screen) a typical model in ***10 seconds or less***

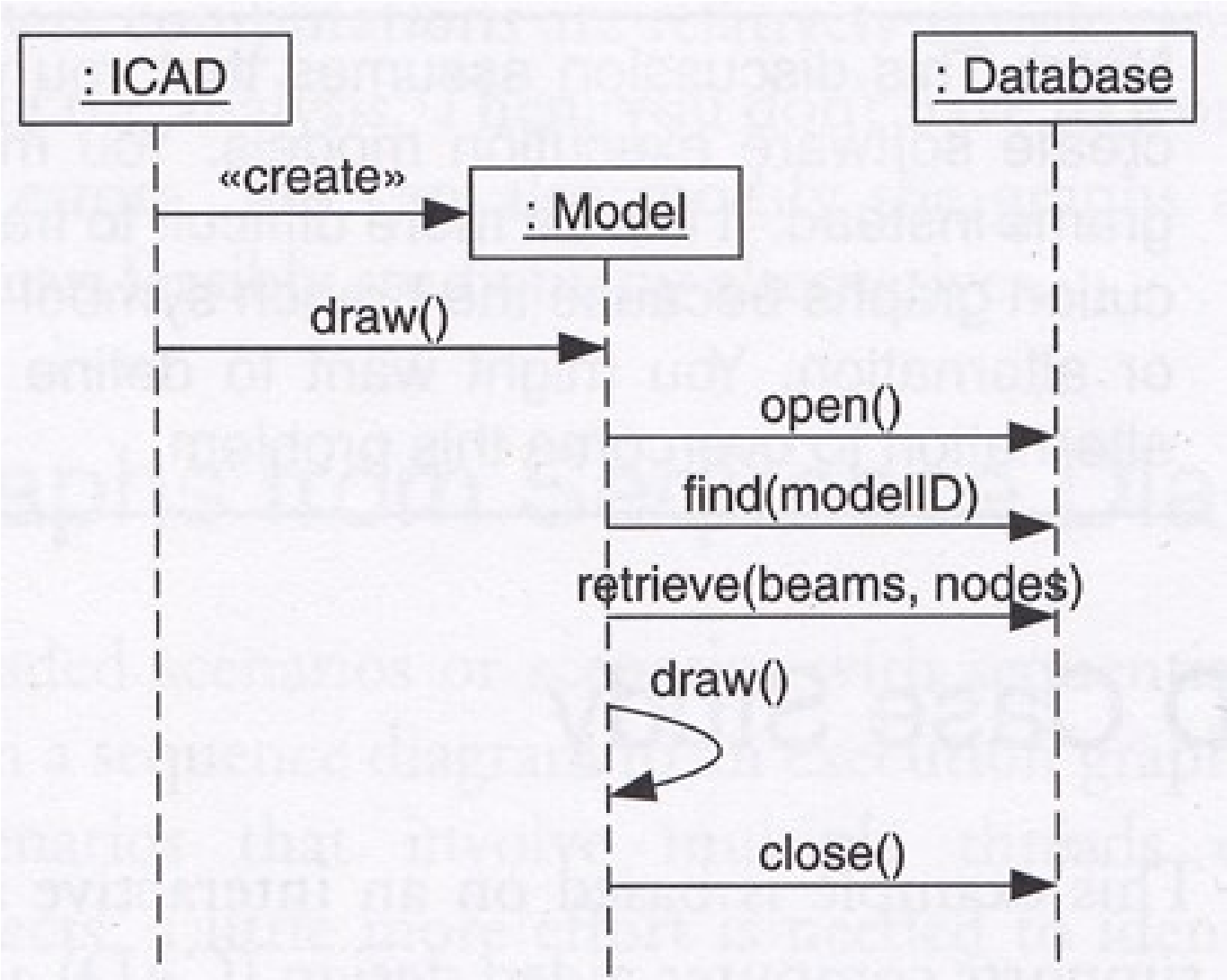


Figure 4-13: The DrawMod Scenario

Architecture 1

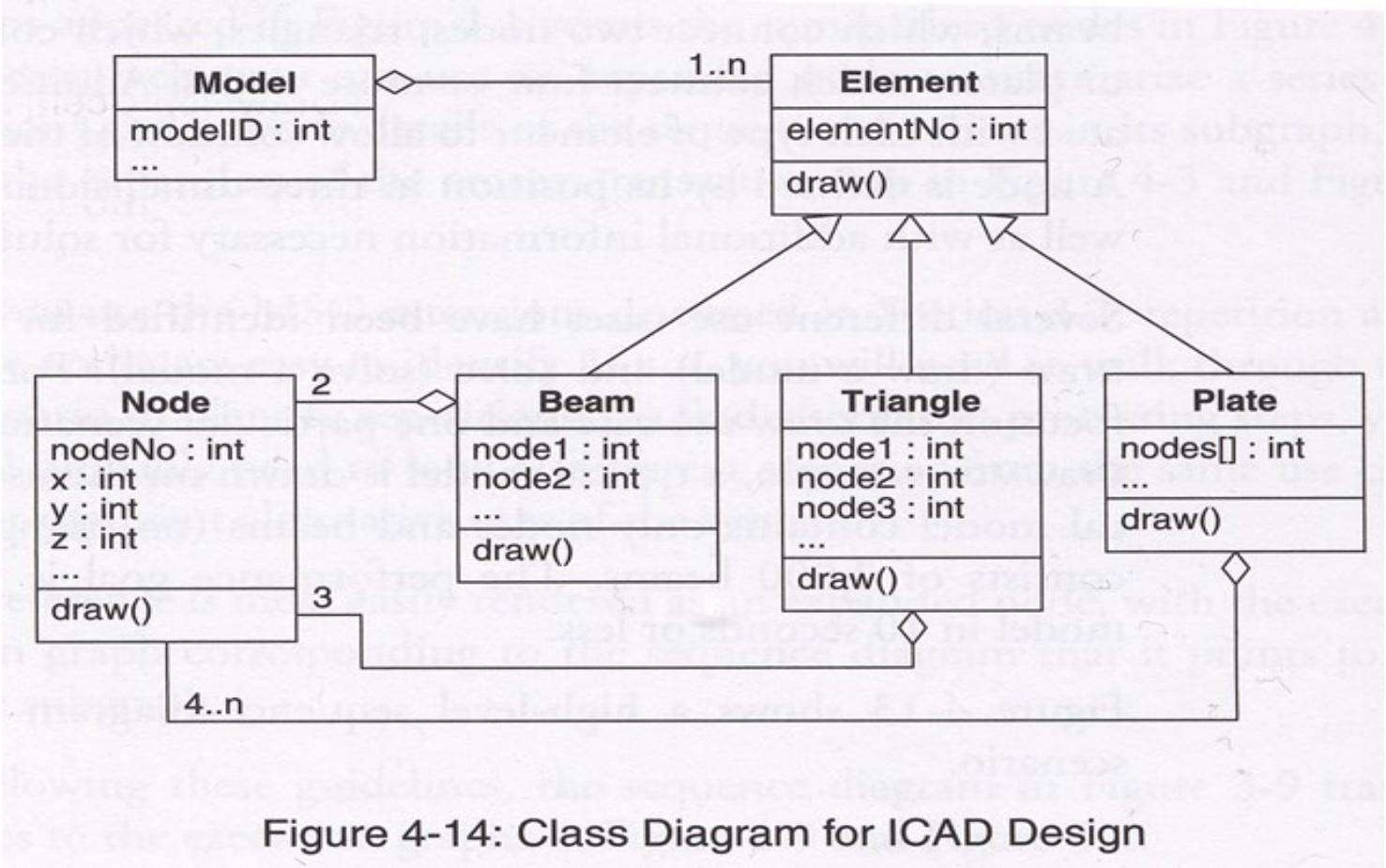


Figure 4-14: Class Diagram for ICAD Design

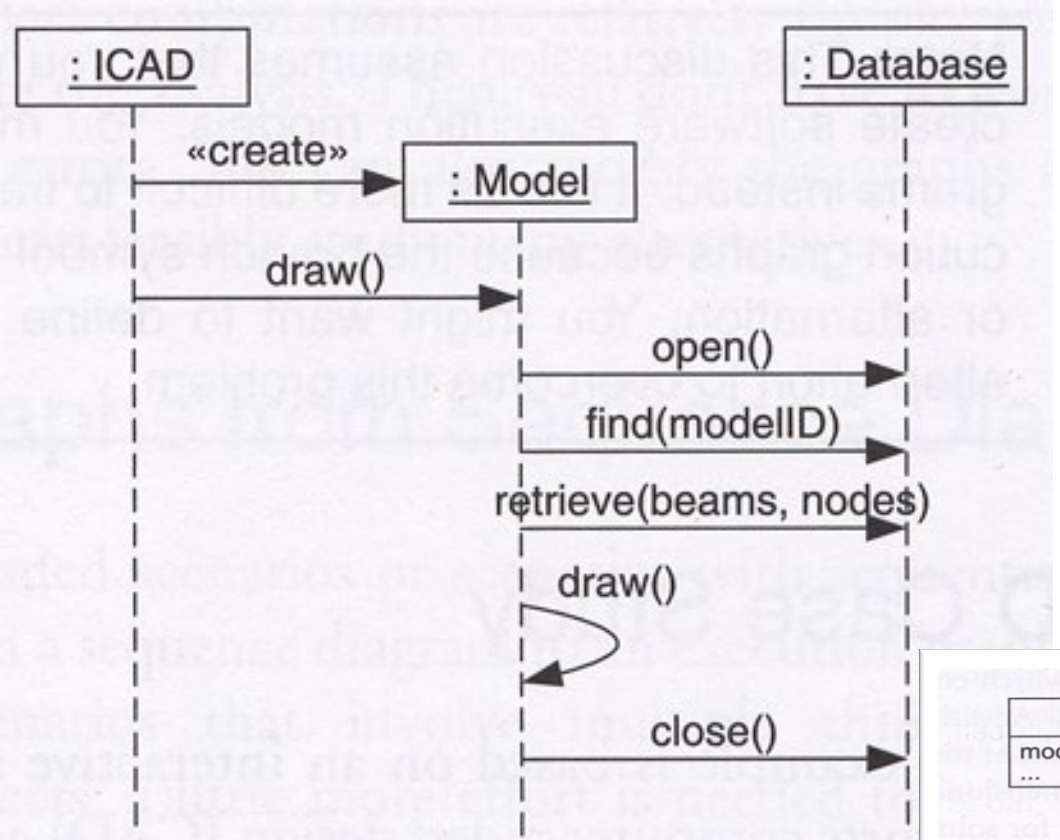


Figure 4-13: The DrawMod Scenari

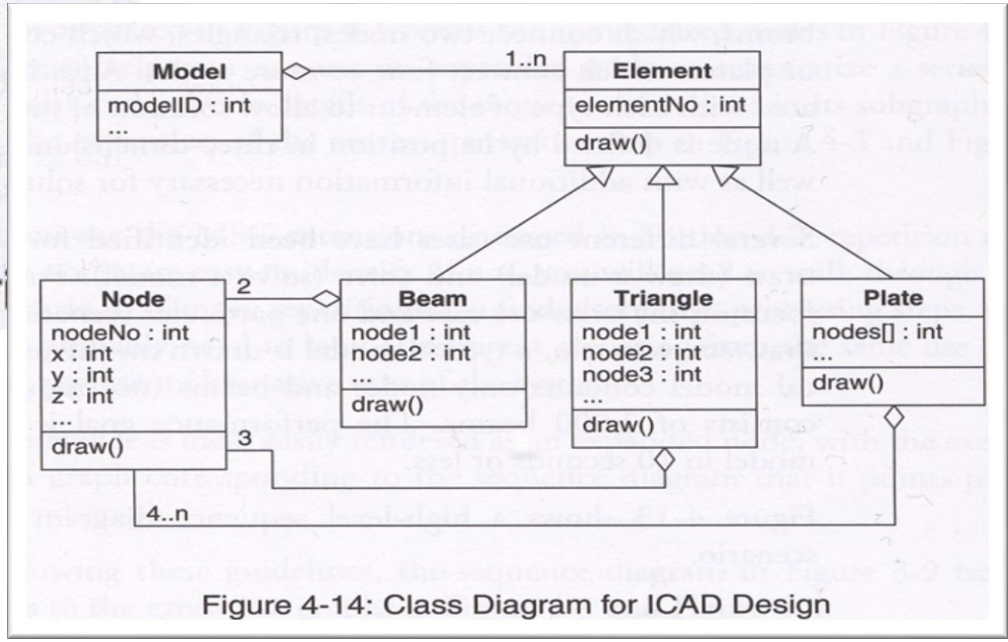


Figure 4-14: Class Diagram for ICAD Design

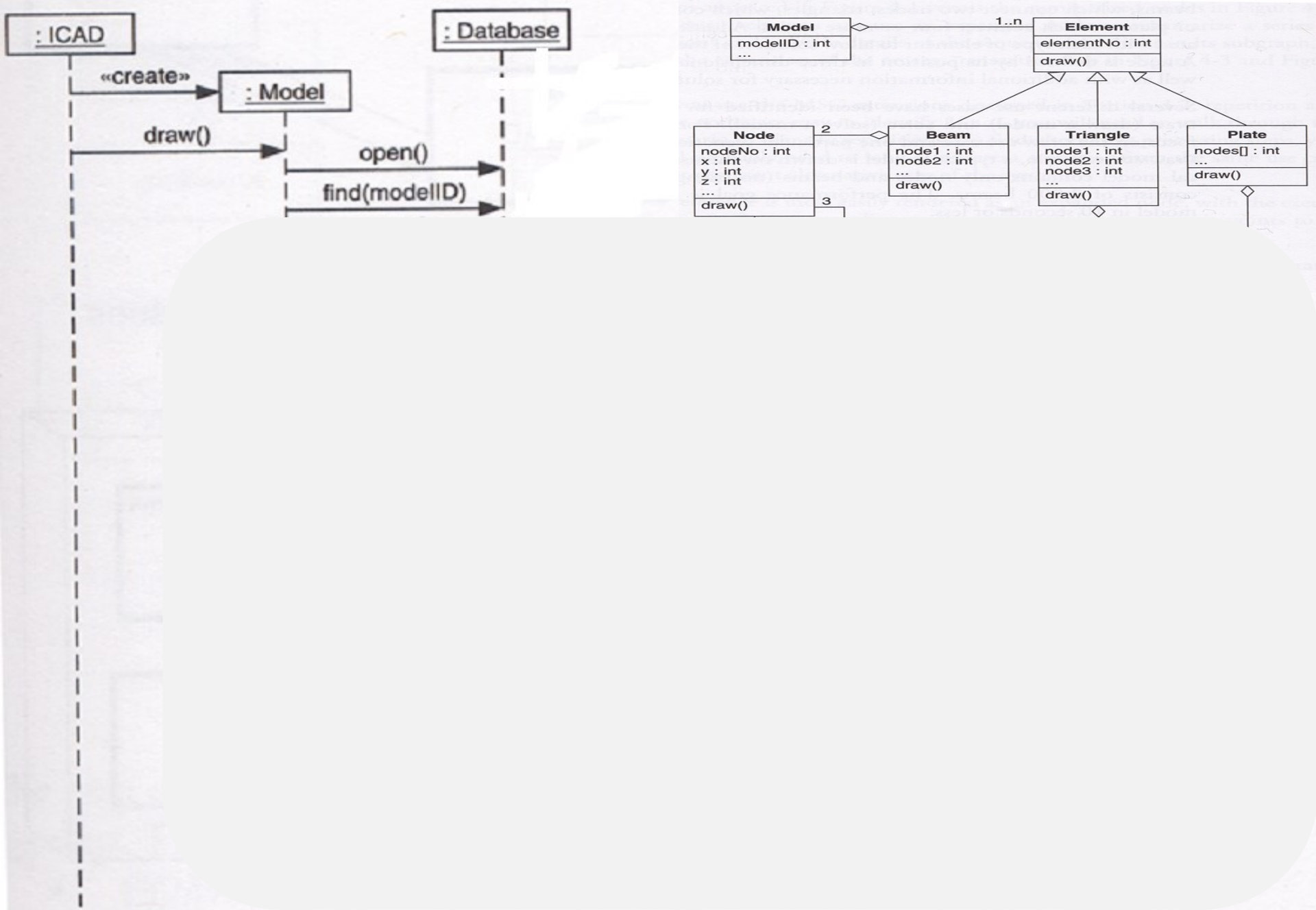
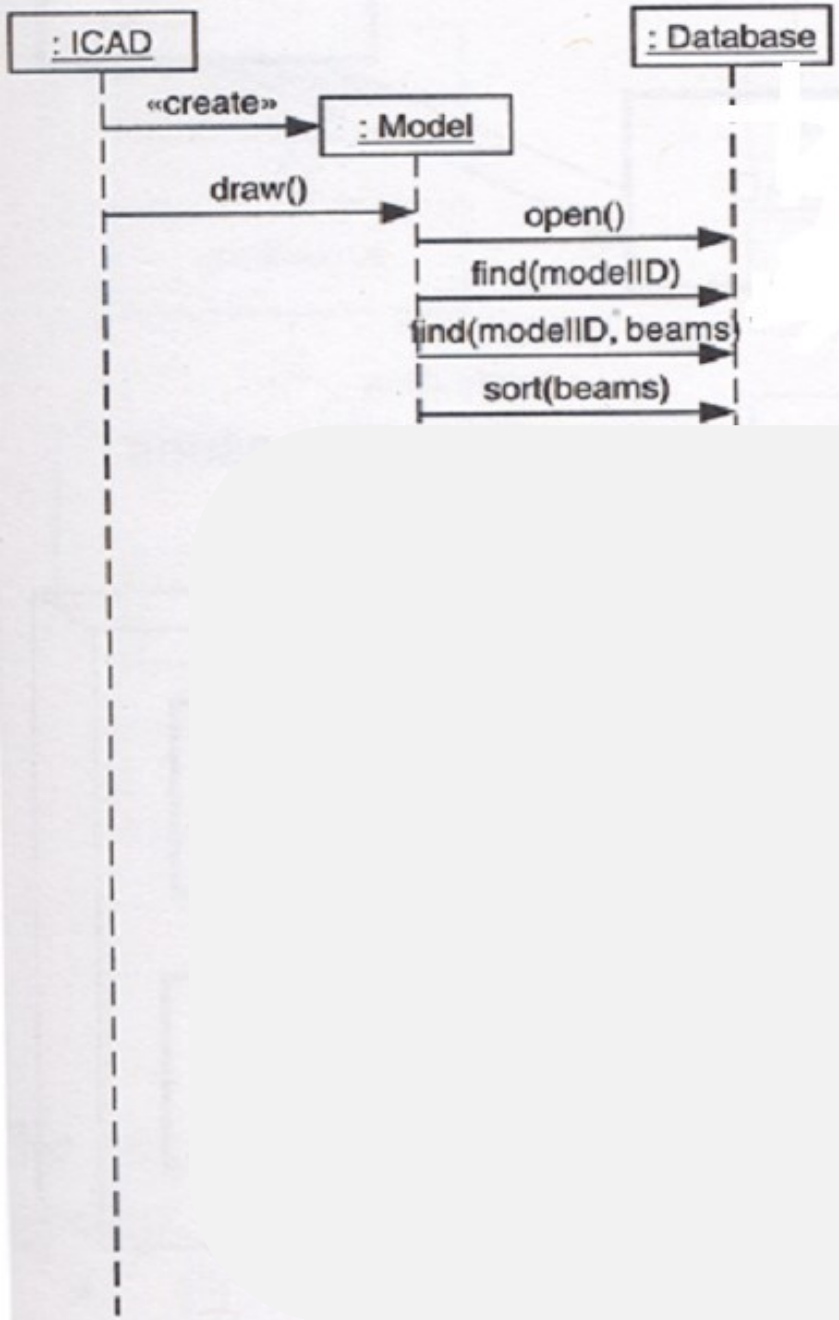


Figure 4-15: Expanded DrawMod Scenario



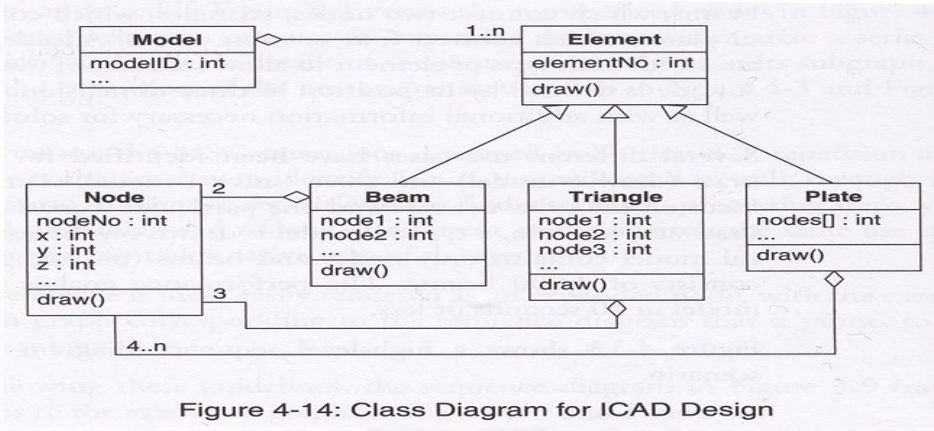


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

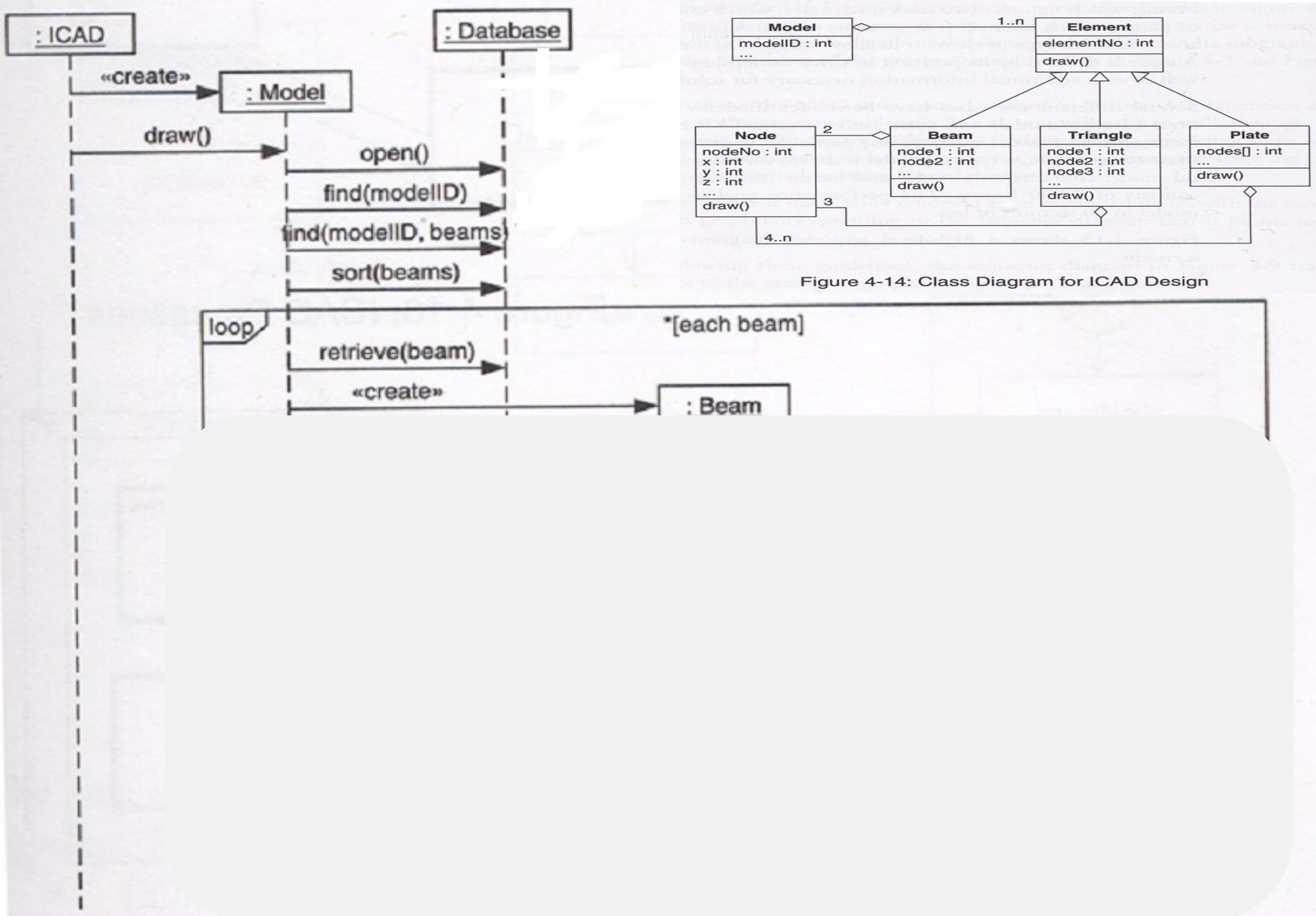


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

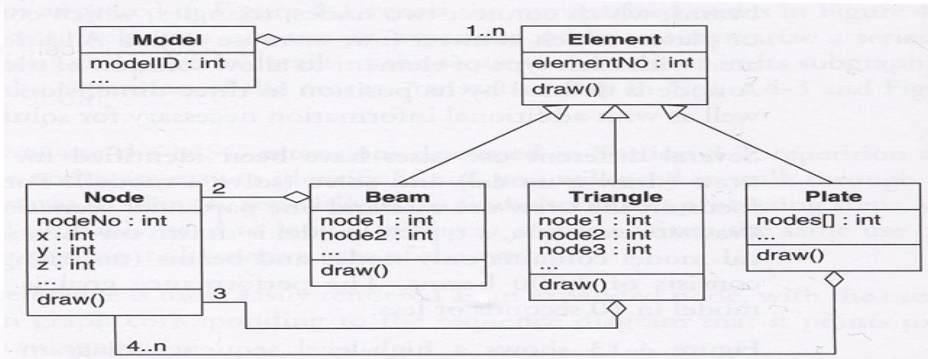


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

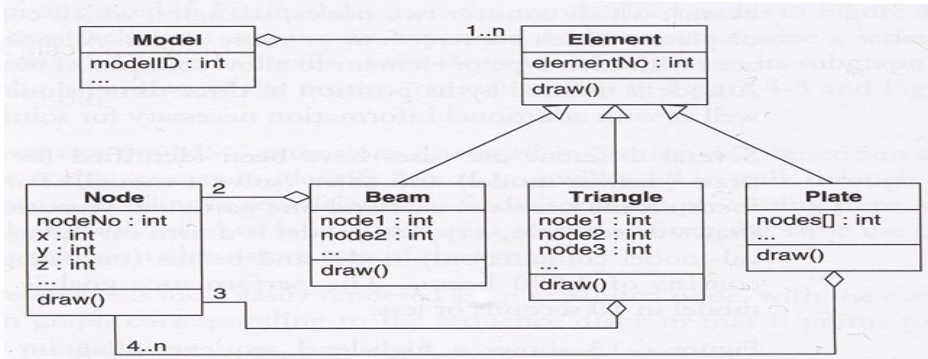


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

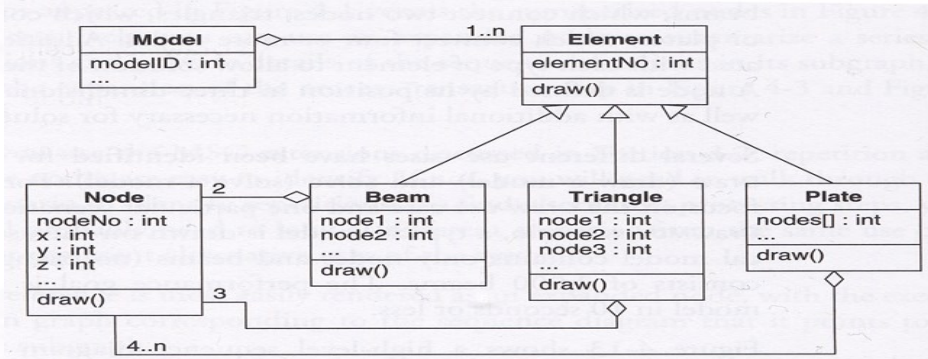


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

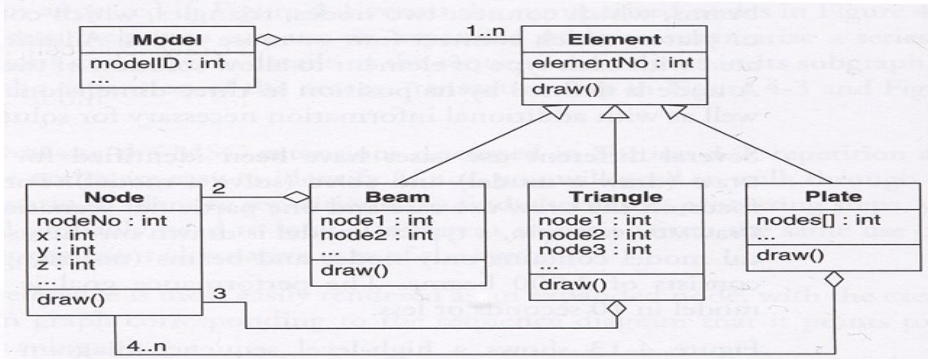


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

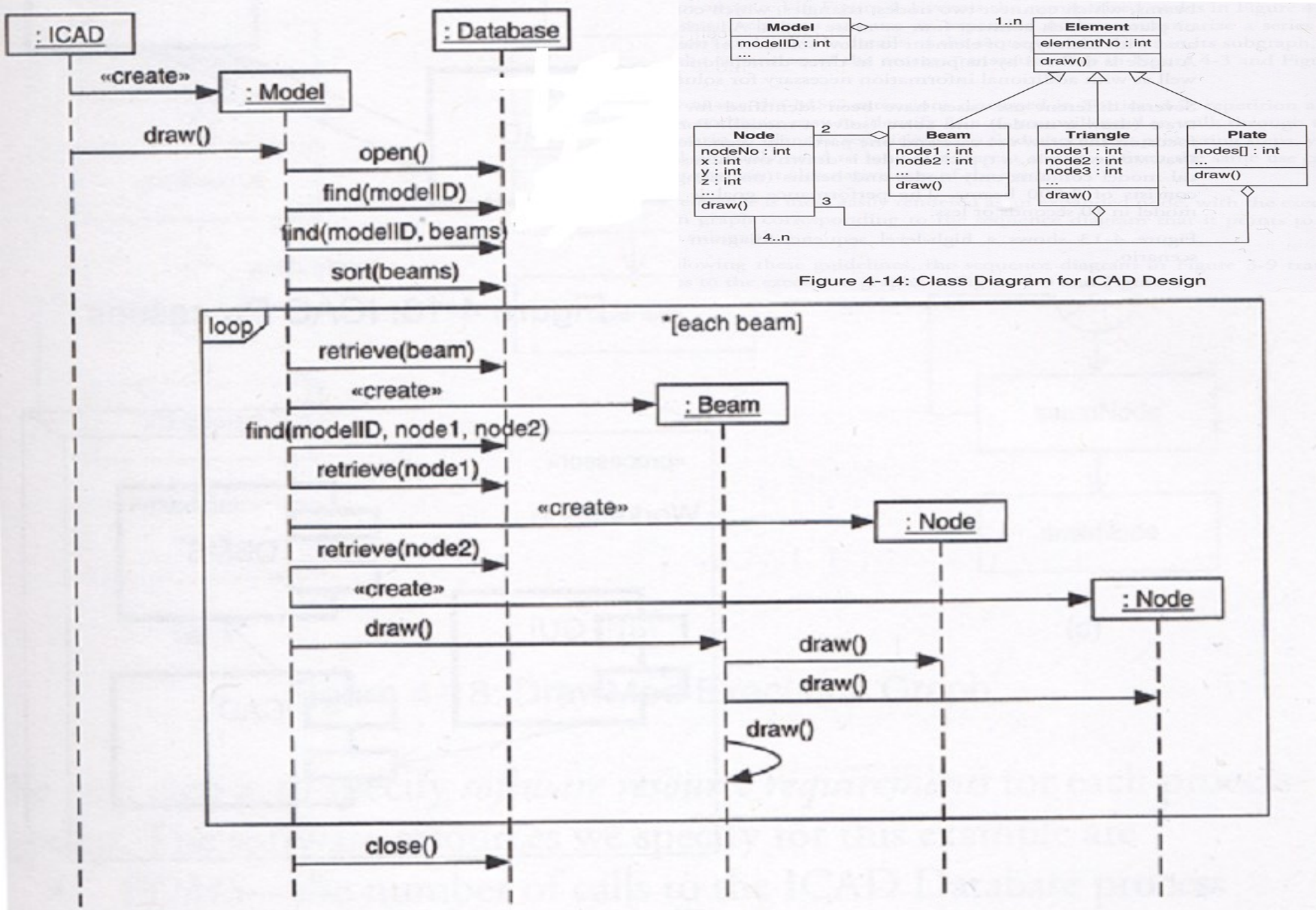


Figure 4-14: Class Diagram for ICAD Design

Figure 4-15: Expanded DrawMod Scenario

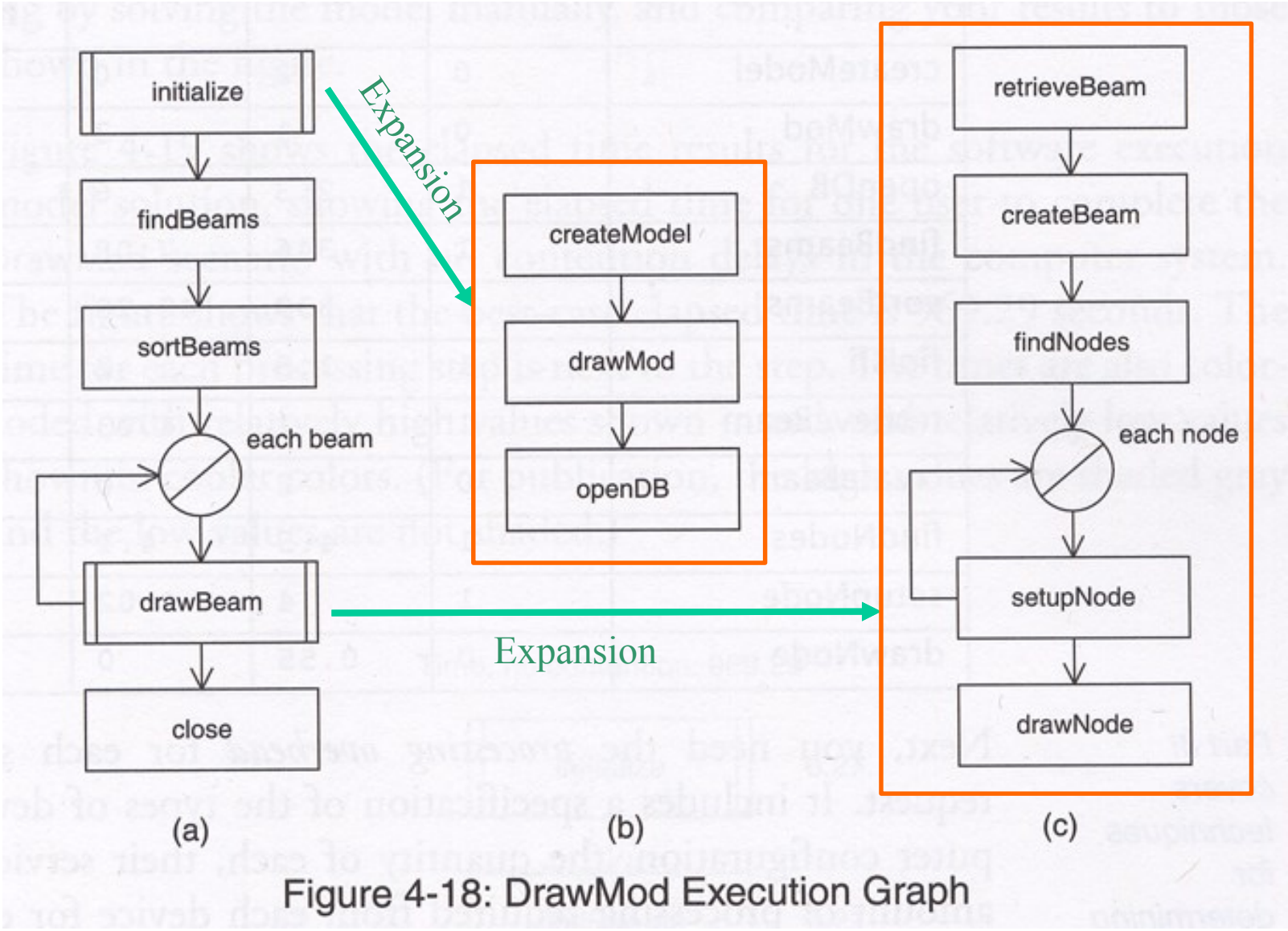


Figure 4-18: DrawMod Execution Graph

Software Resource Requirements

- DBMS – the number of calls to the ICAD Database process
- CPU – an estimate of the number of instructions executed
- I/O – the number of disk accesses to obtain data from the database
- Allocate/Free – the number of calls to the memory management
- Screen – the number of times graphics operations "draw" to the screen

Table 4-3: Values for DrawMod Software Resource Requirements

Processing Step	EDMS	CPU	I/O	Get/ Free	Screen
createModel	0	2	0	0	0
drawMod	0	1	3	2	2
openDB	1	2.3	6	1	0
findBeams	1	346	7.08	0	0
sortBeams	1	339	42.28	2	0
finish	1	1.5	2	1	0
retrieveBeam	1	2	4.03	0	0
createBeam	0	2	0	0	0
findNodes	1	4.5	4.1	0	0
setupNode	1	4	4.02	0	0
drawNode	0	0.55	0	0	1

Table 4-4: Processing Overhead

Devices	CPU	Disk	Display
Quantity	1	2	1
Service Units	K instr.	Phys. I/O	Units

EDMS	0.253	0.002	
CPU	1		
I/O	0.1	1	
Get/Free	0.1		
Screen	0.05		1

Service Time	0.000005	0.03	0.001

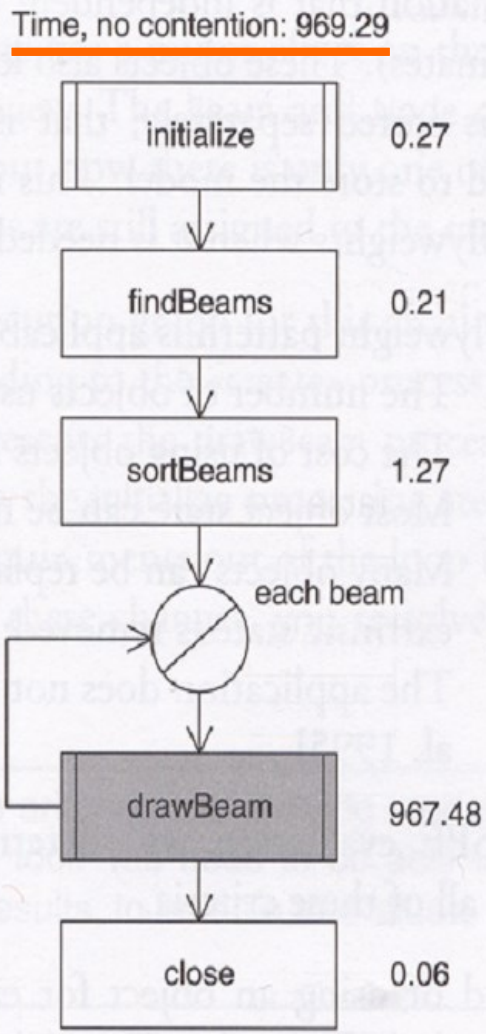


Figure 4-19: Best-case Elapsed Time for DrawMod Architecture 1

Architecture 1

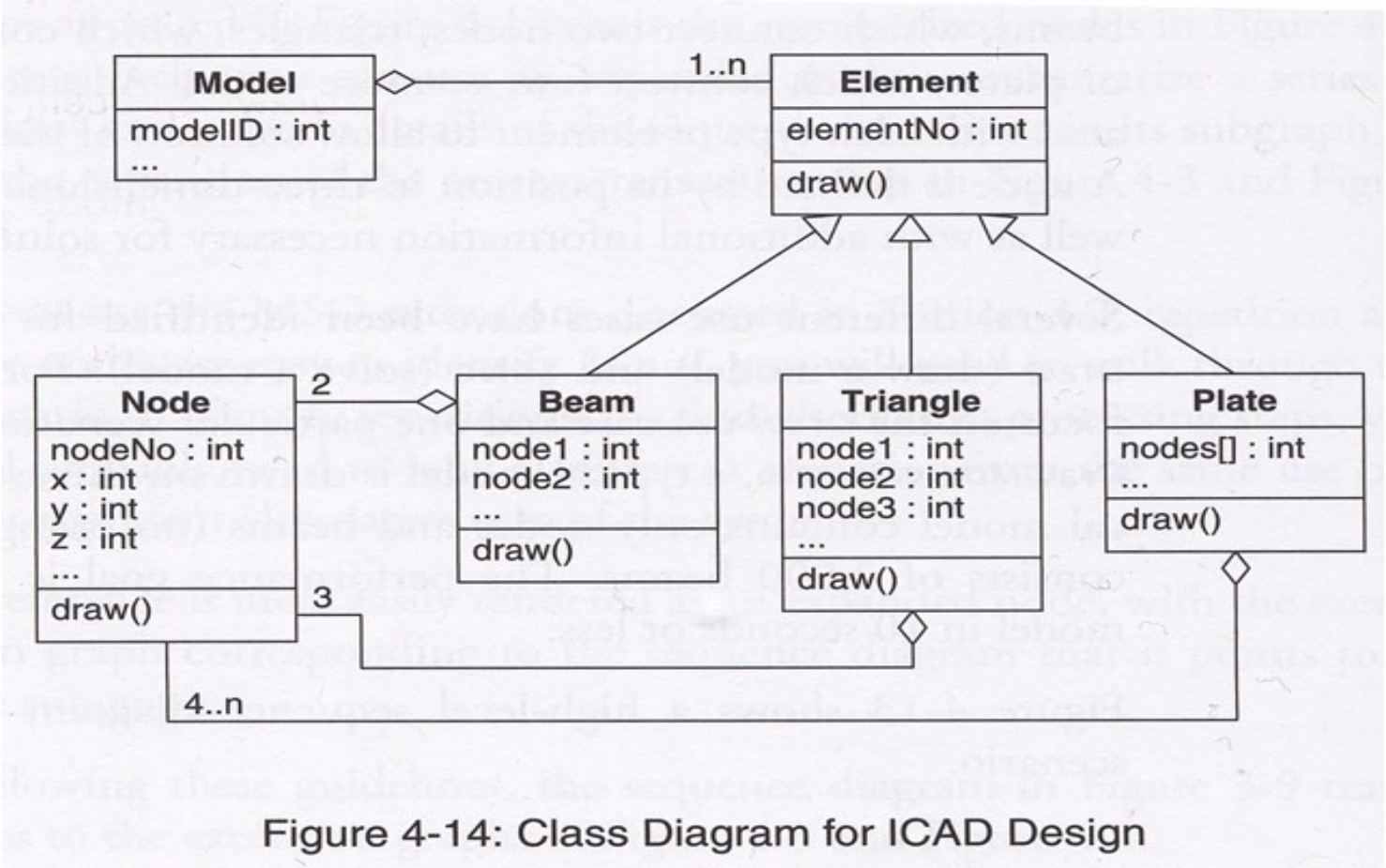
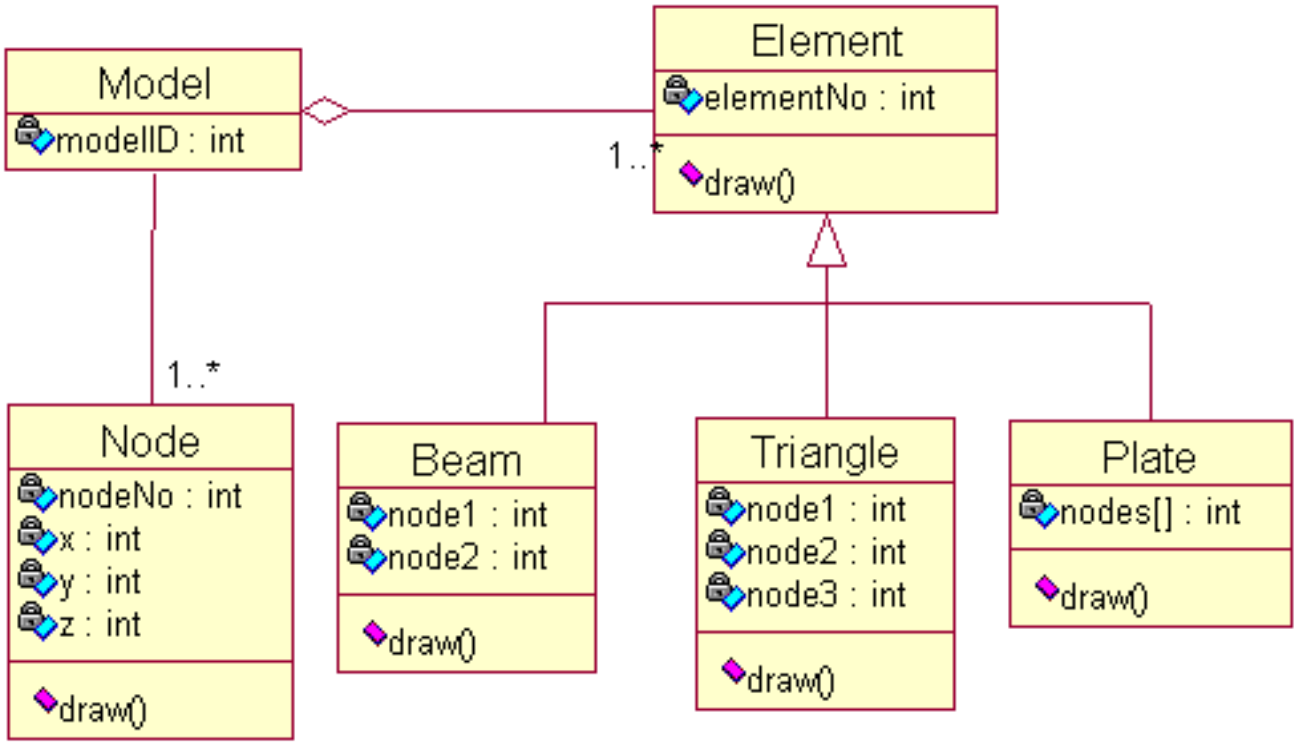


Figure 4-14: Class Diagram for ICAD Design

Architecture 2



969.29 → 969.23

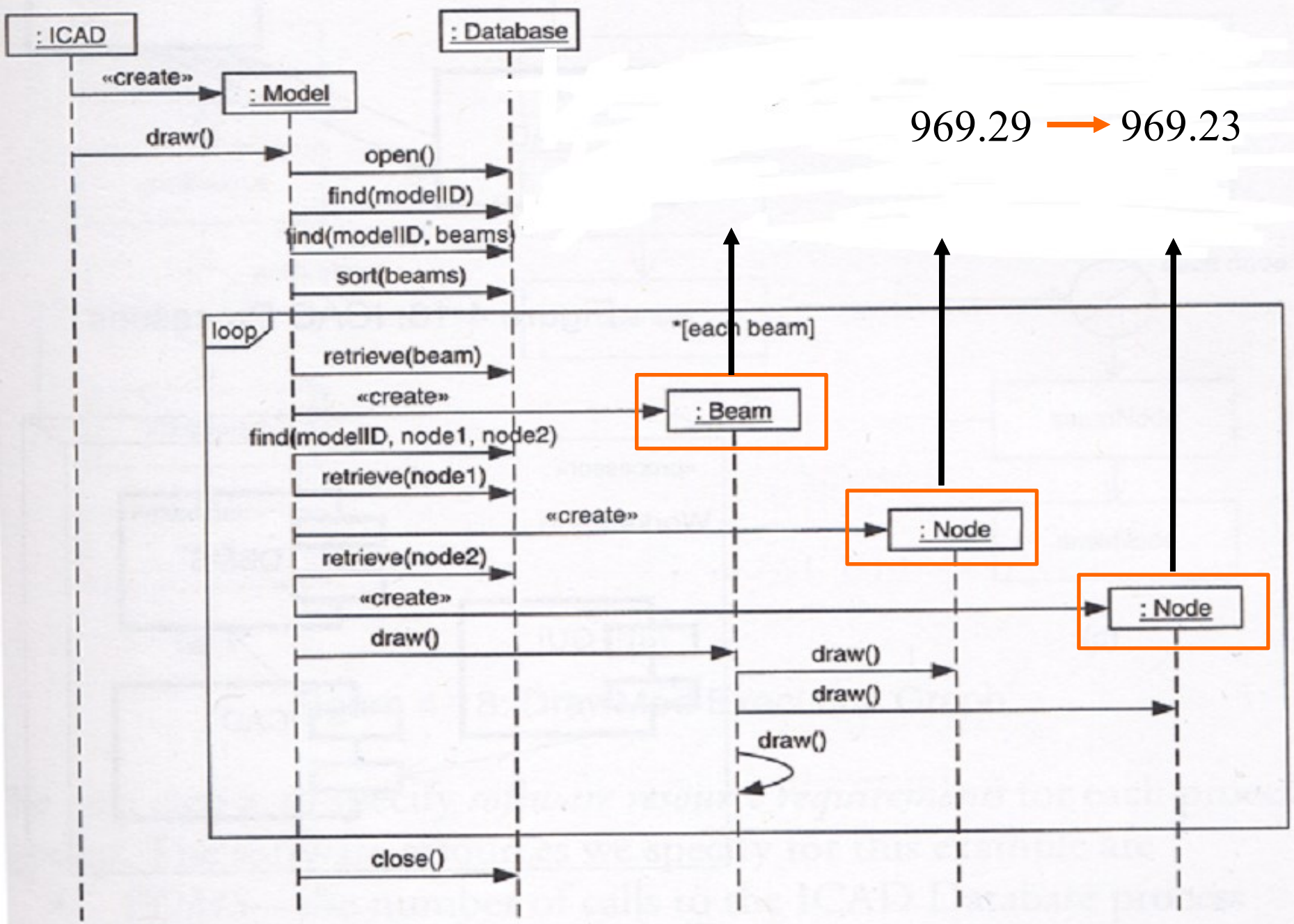


Figure 4-15: Expanded DrawMod Scenario

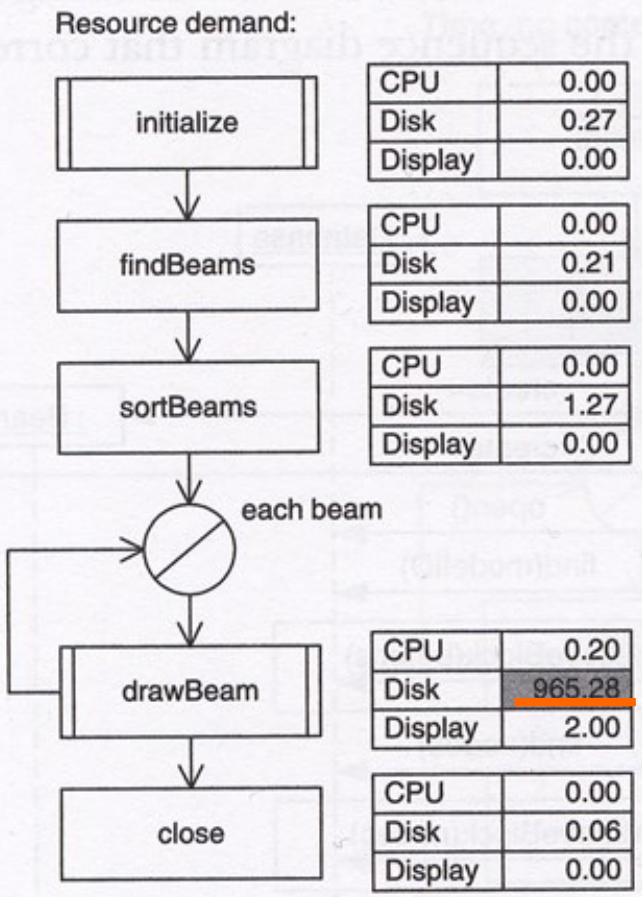


Figure 4-21: Resource Demand Results for Architecture 1

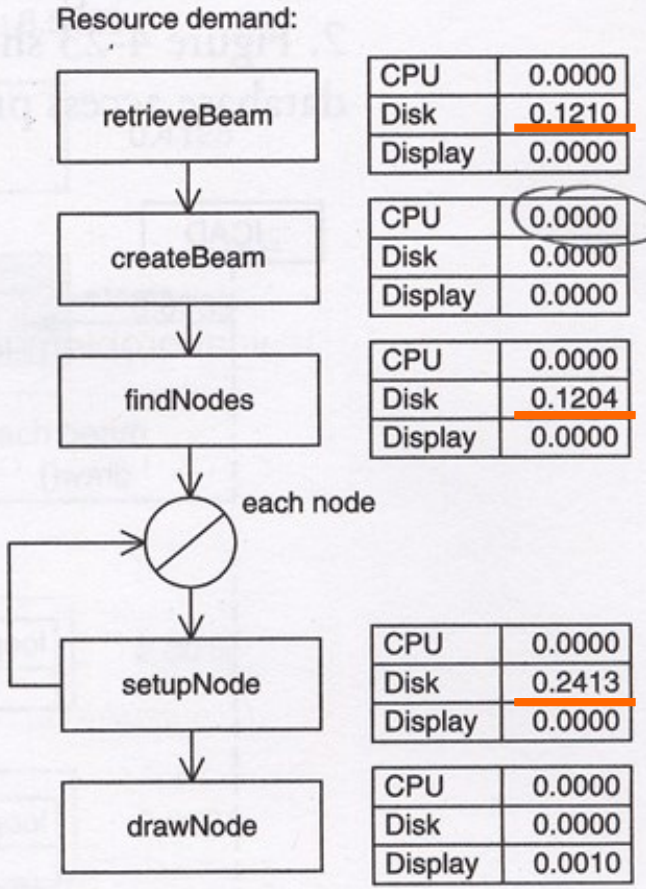


Figure 4-22: Resource Demand for drawBeam Submodel in Architecture 1

Winning Architecture 3

Resolving the bottleneck

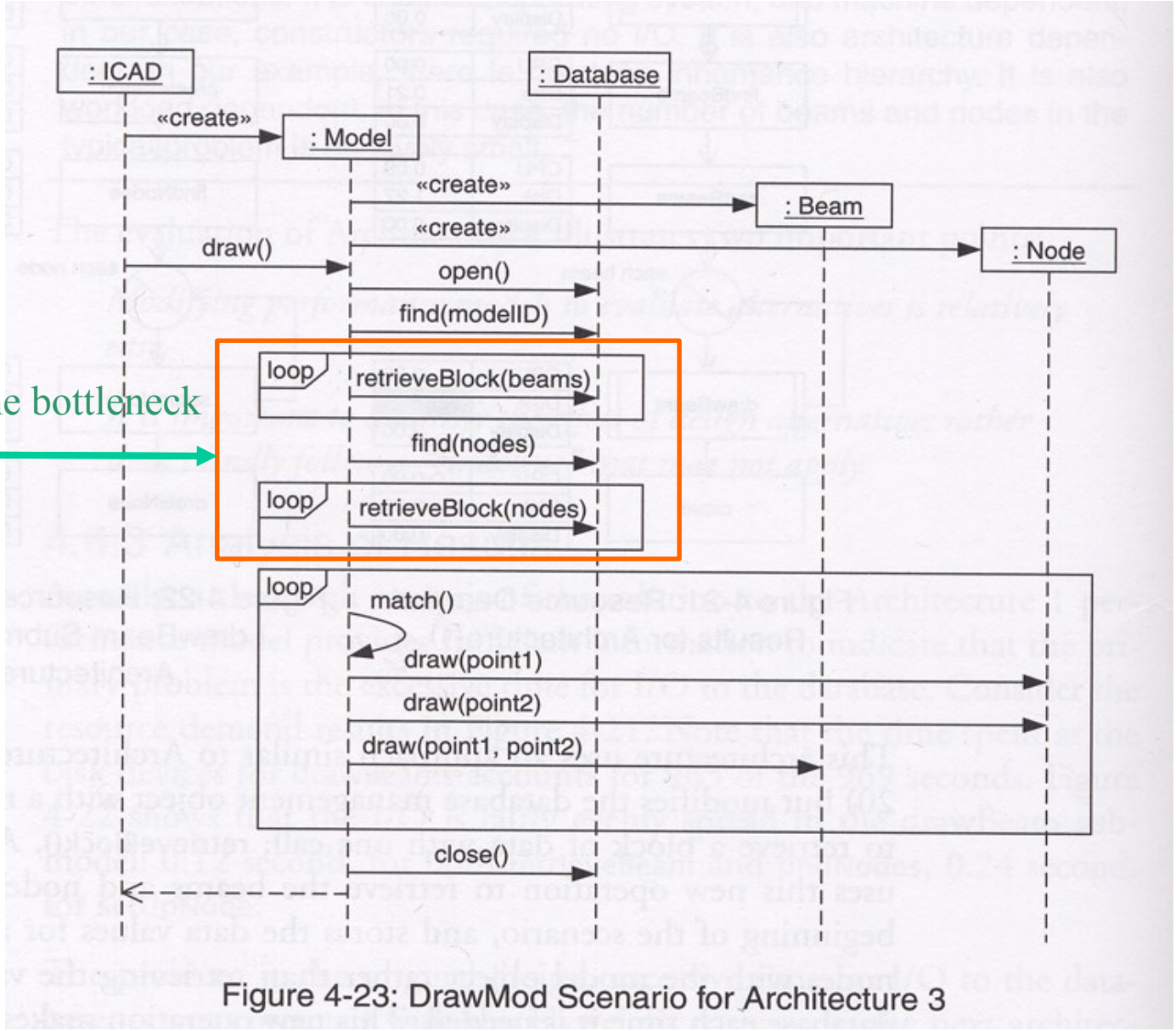


Figure 4-23: DrawMod Scenario for Architecture 3

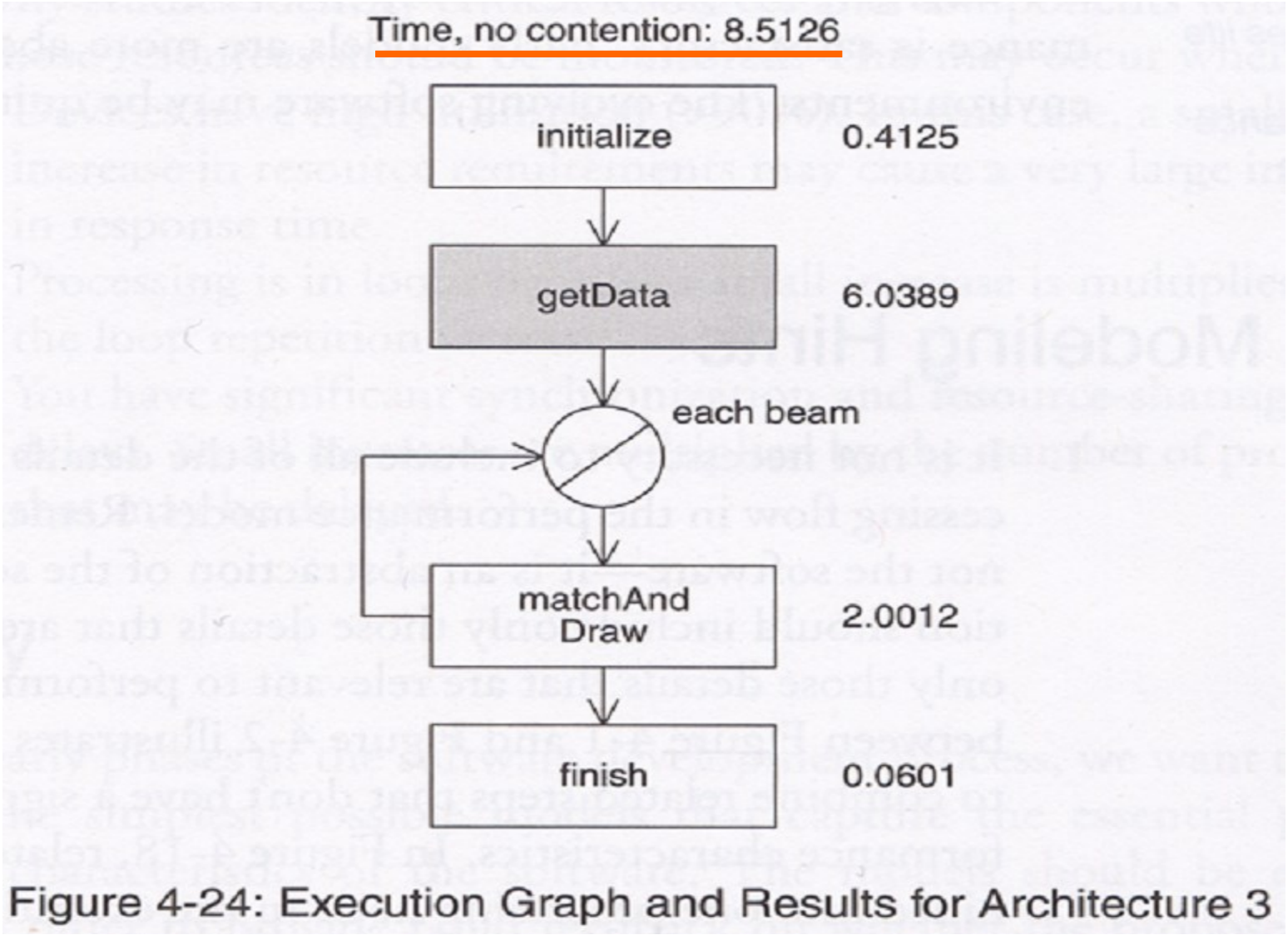


Figure 4-24: Execution Graph and Results for Architecture 3

Modeling Hints

- It is not necessary to include all of the details of the software's processing flow in the performance model
- Use hierarchy to help make your models easier to understand and modify
- Use best- and worst-case estimates of resource requirements to help compensate for uncertainty early in the process
- Study the sensitivity of the performance results to the input parameters