

## **Critical Thinking Assignment 4: UML Activity Diagram**

Alexander Ricciardi

Colorado State University Global

CSC470: Software Engineering

Dr. Vanessa Cooper

January 12, 2025





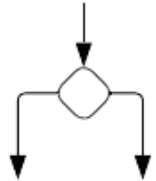
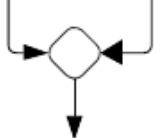
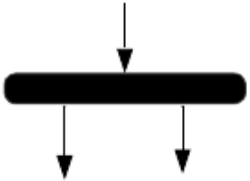
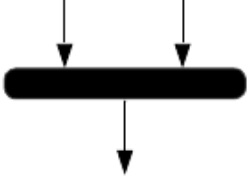

## **Critical Thinking Assignment 4: UML Activity Diagram**

Unified Modeling Language (UML) activity diagrams describe the workflow behavior and dynamic aspect of systems (Thanakorncharuwit et al. 2016). They are similar to flowcharts in that they depict the flow of activities (IBM, 2021). However, activity diagrams can also illustrate a system's multithread components, showing parallel or concurrent flows, as well as alternate flows. The diagrams are used in software engineering and business modeling to describe and design the dynamic aspect of systems. This essay analyzes the workflow and dynamic behavior of a customer's interaction with a restaurant system, focusing on the dine-in customer experience; by providing an activity diagram and analysis of it. The analysis examines the diagram's partitions, multithreading elements, and decision points. Additionally, the essay discusses how multiple threads help to optimize workflows.

### **UML Activity Diagrams Overview**

In the context of Software Engineering (SE), UML activity diagrams are used to show any flow or process in a system (Unhelkar, 2018). They are a great tool for modeling business processes or workflows within the organization, the flow within a use case by creating a visual representation of it based on the documentation of that use case, dependencies between use cases, and user interface navigation (storyboard) of an application. In other words, activity diagrams can be used in all three SE modeling spaces (MOPS, MOSS, and MOAS). However, they are mostly used in the Model of Process Space (MOPS) to model process flows and the Model of Solution Space (MOSS) to model multithreaded and multitasked processes. The table below, Table 1, describes some of the components that can be found in activity diagrams. These components consist of nodes representing different process controls that include activity states, decisions, forks, joins, and objects (Thanakorncharuwit et al. 2016).

**Table 1***UML Activity Diagram Components*

Symbol	Name	Description
	Start	Represents the starting point of an activity
	Activity	Represents the activities of the process
	Control Flow	Represents the flow of control from one action to the other
	Activity Final Node	End of an activity
	Decision Node	Represents a conditional branching point with one input and multiple outputs (yes, no)
	Merge Node	Represents the merging of several flows. Several inputs, but one output.
	Fork	Represents a flow branching into two or more parallel flows
	Join (Parallel Merge)	Represent two or more parallel flows branching into one flow
	Comment	Adds comments

*Note:* The images are from a Lucichart App. of an activity diagram “Activity Diagram Restaurant Dine-in Customer” by Ricciardi (2025).

## Strengths and Weakness of UML Activity Diagrams

Activity diagrams provide several advantages; however, they also have disadvantages, see Table 2 for a list of those advantages and disadvantages. In the context of SE, the main strength of the diagrams is their capacity for modeling process flows. On the other hand, their main weakness is that they have minimal structural characteristics and do not directly indicate how a system or its requirements are organized and prioritized (Unhelkar, 2018).

**Table 2**

*Strengths and Weakness of UML Activity Diagrams*

UML Activity Diagrams	
Strengths	Weaknesses
Model the flow within and across use cases.	Provide minimal structural information about a system.
Visual representation of a use case internal flow.	Do not directly show how system requirements are organized or prioritized.
Detailed illustration of flows.	Do not offer a complete picture of a system's requirements (unlike use case diagrams).
Can illustrate multiple concurrent flows using forks and joins.	Struggle in organizing and managing a large number of requirements.
Depict decisions and alternative paths with decision points.	Complex use cases can result in overly complicated diagrams; breaking them down into smaller diagrams is recommended.
Maps actors to the activities they perform using partitions.	Should only be used when there is a need to show dependencies between activities.

Bridge the gap between use cases and sequence diagrams.	In the solution space, they have limited value unless the system uses multithreading or multitasking.
Easy to understand, even for individuals without a technical background.	May be confused with state machine diagrams (in earlier UML versions) due to similar notations.
Can be used to enhance readability and understanding.	May be confused with data flow diagrams (DFDs); activity diagrams focus on workflow, while DFDs focus on data flow.
Can be used for user-level documentation, such as user manuals and help files.	Different types of processes cannot be captured in a single activity diagram; multiple diagrams are needed, linked by notes.
Facilitate workshops, especially when participants are new to modeling.	Even though they can describe dynamic aspects of a system, activity diagrams do not represent time, therefore they are not considered dynamic representations of a system.
In business process engineering, they are used to document and optimize existing processes.	
In software, they can also be used in documenting software development.	

*Note:* data from “Chapter 7 — Activity Diagrams, Interaction Overview Diagrams, and Business Process Models. Software Engineering With UML” by Unhelkar (2018).

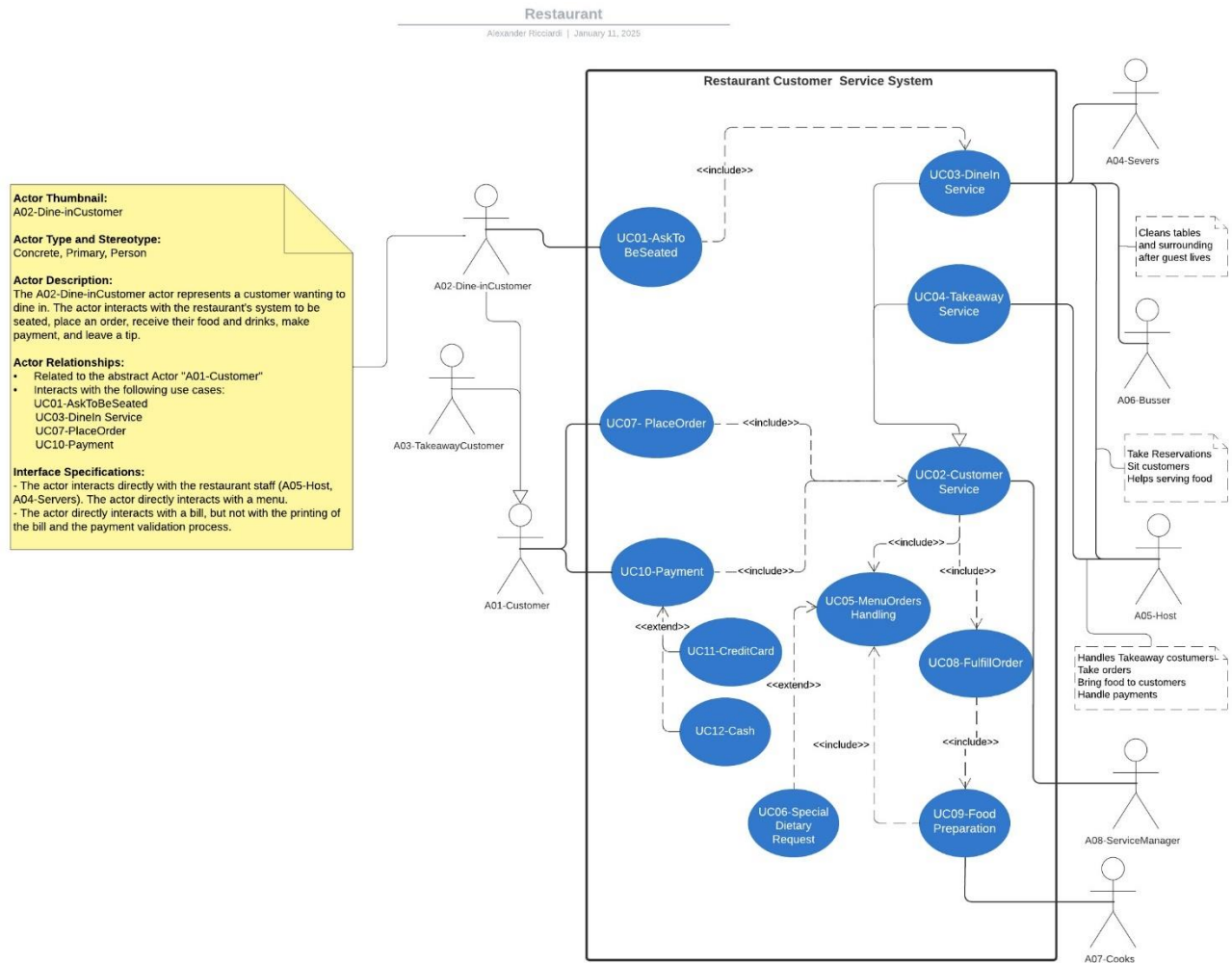
### **UML Activity Diagram Example**

Activity diagrams are typically created after UML use case diagrams and actor/use case documentation (Vpadmin, 2023). In the context of SE, use case diagrams are part of the early stage of software development, as they play an important in capturing and modeling a system's behaviors (Ricciardi, 2025). They capture a system use cases and actors. These capture use cases and actors and the documentation associated with them can be used as a base for developing

activity diagrams that illustrate the workflow of each use case or the overall workflow of a business process from the perspective of one or several actors.

**Figure 1**

### *UML Restaurant Use Case Diagram*



*Note:* from "UML Use Case Diagrams: A Restaurant System Case Study" by Ricciardi (2025).

Modified.

### **Base UML Use Case Diagram Example**

The use case diagram from "Restaurant Customer Service System Diagram" (Ricciardi, 2025), see Figure 1, and the A02-Dine-inCustomer actor documentation associated with it can be

used as a basis to create an activity diagram. Below is the A02-Dine-inCustomer actor documentation from the “UML Restaurant Use Case Diagram”:

- Actor Thumbnail: A02-Dine-inCustomer
- Actor Type and Stereotype: Concrete, Primary, Person
- Actor Description:

The A02-Dine-inCustomer actor represents a customer wanting to dine in. The actor interacts with the restaurant's system to be seated, place an order, receive their food and drinks, make payment, and leave a tip.

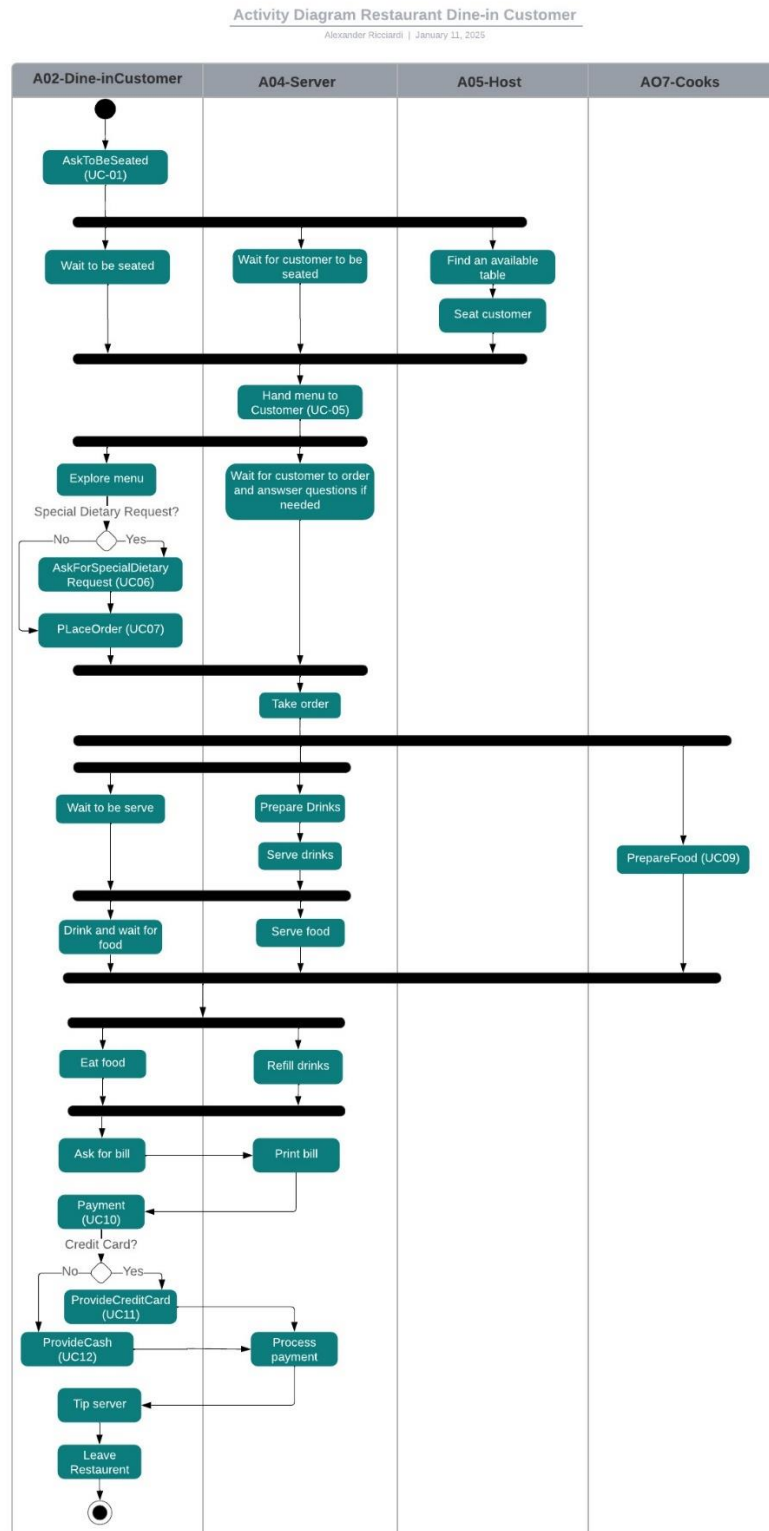
- Actor Relationships:

- Related to the abstract Actor “A01-Customer”
- Interacts with the following use cases:
  - UC01-AskToBeSeated
  - UC03-DineIn Service
  - UC07-PlaceOrder
  - UC10-Payment

- Interface Specifications:

- The actor directly interacts with the restaurant staff (A05-Host, A04-Servers). The actor directly interacts with a menu.
- The actor directly interacts with a bill, but not with the printing of the bill and the payment validation process.

Using the example above a business process activity diagram of the restaurant system from the perspective of a dine-in customer can be created, see Figure 2.

**Figure 2***Restaurant Dine-in Customer Activity Diagram*



## The Activity Diagram Example Overview

The “Restaurant Dine-in Customer Activity Diagram,” Figure 2, is an illustration of a restaurant business process from the perspective of a dine-in customer. In other words, the diagram provides a visualization of the workflow of a dine-in customer at a restaurant, from the moment they enter the establishment to when they leave. This activity diagram is based on the “Restaurant Customer Service System” use case diagram, Figure 1, and the “A02-Dine-inCustomer” actor documentation. The following list breaks down to diagram by component to provide a detailed overview of it:

### - Partitions and Actors:

The diagram uses partitions (swimlanes) to partition activities by actors.

- A02-Dine-inCustomer: The primary actor, representing the customer who is dining in.
- A04-Server: Represents the waiter or waitress responsible for serving the customer.
- A05-Host: Represents the staff member responsible for greeting and seating customers.
- A07-Cooks: Represents the kitchen staff responsible for preparing the food.

### - Workflow and Activities:

The diagram depicts the flow of activities, starting with the “A02-Dine-inCustomer” asking to be seated, “AskToBeSeated (UC-01).” Then the “A05-Host,” finds an available table, while the “A02-Dine-inCustomer” and “A04-Server” wait, after a table is found, the host seats the customer. Then the “A04-Server” hands the menu to the customer, answers questions, waits for the customer to pick drinks and food from the menu, and then takes their order. Note that The decision point “Special Dietary Request?” allows an alternate flow if the customer has specific dietary needs.

### - Multithreading, Parallelism, or Concurrency:

The fork and join nodes illustrate multithreading and concurrent activities. For example:

- Fork-3: After taking the order the the “A07-Cooks” “PrepareFood (UC09).”
  - Fork-4: While the “A02-Dine-inCustomer” “waits to be served” and “A04-Server” “prepares the drinks” and “serves the drinks.”
  - Join-4: After the “A04-Server” “serves the drinks” the “A02-Dine-inCustomer” “drinks and waits for the food, while the “A07-Cooks” “PrepareFood (UC09).”
- Join-3: After “A04-Server” “serves the food” Fork-5 starts, where “A02-Dine-inCustomer” “eats the food” and “A04-Server” “refills the drinks.”

Note that the Fork-3 and Join-3 are encapsulated between the Fork-4 and Join-4 demonstrating how a multithread can be encapsulated in other multithread.

- Decision Points and Alternate Flows:

Decision points represent choices and alternate flows within the process. The “Special Dietary Request?” decision allows for a separate path where the customer can

“AskForSpecialDietaryRequest (UC06).” The “Credit Card?” and “Payment successful?”

decision points within the payment process (UC10) handle different payment scenarios,

- Error Handling:

The diagram does not handle errors, this can be implemented in different activity diagrams based on specific use cases.

### **Multithread and Workflow Optimization**

In the context of the "Restaurant Dine-in Customer Activity Diagram," multithreads are represented by the fork and join components of the diagram, they illustrate the concurrent execution of different activities as shown in the activity diagram example overview.

Multithreading enhances the efficiency, responsiveness, and resource utilization of a system.

This is true for both business process modeling and software engineering. For instance, the concurrent processing enabled by multithreading reduces the overall time required to complete a

process and reduces bottlenecks as activities are not performed sequentially. Additionally, multithreading makes a system more responsive by allowing actors (users, clients) to interact with the system in parallel. Furthermore, it allows for better utilization of available resources by allowing multiple tasks to be performed concurrently, maximizing the use of resources such as workers' productive time or power. Thus, it is important to optimize workflows by implementing, when possible, multithreading. Examples of applications where multithreading is valuable include operating systems, where multithreading is extensively implemented to manage various system processes and user applications, and e-commerce applications, where it is implemented to handle multiple customer requests concurrently.

### **Conclusion**

Whether in software engineering or business process modeling, UML activity diagrams are used to describe the workflow behavior and the dynamic aspects of systems. As shown in the "Restaurant Dine-in Customer Activity Diagram," activity diagrams can model business processes or workflows within an organization or system, by utilizing elements such as partitions, decision points, and fork and join nodes. They can depict the flow of activities, actor responsibilities, alternate paths, and concurrent processes. Additionally, they can be used to optimize workflow by identifying multithreading opportunities; therefore, improving the overall efficiency of a system. UML activity diagrams are a powerful tool for not only understanding and documenting systems' behavior but also for improving and optimizing them.

## References

- IBM (2021, March 03). Activity diagrams. *IBM Rational Software Architect documentation*.  
IBM Documentation. <https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-activity>
- Thanakorncharuwit, W., Kamonsantiroj, S., & Pipanmaekaporn, L. (2016). Generating test cases from UML activity diagram based on business flow constraints. *Proceedings of the Fifth International Conference on Network, Communication and Computing*, 155–160.  
<https://doi-org.csuglobal.idm.oclc.org/10.1145/3033288.3033311>
- Ricciardi, A. S. (2025, January 2). UML Diagrams: A Guide for Software Engineers - Level up Coding. *Medium*. <https://medium.com/gitconnected/uml-diagrams-a-guide-for-software-engineers-71220ffb775f>
- Unhelkar, B. (2018). Chapter 7 — Activity diagrams, interaction overview diagrams, and business process models. *Software engineering with UML*. CRC Press. ISBN 9781138297432
- Vpadmin. (2023, October 11). *Unraveling Use cases: A Step-by-Step Guide to Elaboration through activity Diagrams - Visual Paradigm Guides*. Visual Paradigm Guides.  
<https://guides.visual-paradigm.com/unraveling-use-cases-a-step-by-step-guide-to-elaboration-through-activity-diagrams/>