e-Build

On-Demand Manufacturing System
Phase One: FSM

Description of the *e-Build* System

You are charged with the task of building a software system that manages on-demand manufacturing of auto parts. Your system starts in a listening state waiting for incoming orders. Each order requests only one type of auto-parts, but may request multiple replicas of the same part in the same order. After being received, an order is next processed to obtain, among other things, valid payment information such as a valid credit card. Next, the factory lines are commissioned with the task to manufacture the required number of replicas of the specific auto part being ordered. Finally, all the manufactured items are shipped together as one package to the client with delivery confirmation. The payment method supplied by the client is charged at the time of shipping. Once a successful delivery is confirmed, the order is considered closed, and the system is now moved to wait for new orders. At this phase of the project, the system can handle one order at a time from beginning till closure before any new order can be accepted. At this phase of the project, there is only one factory line to manufacture all the replicas of the auto part being ordered.

The system must implement the state diagram shown in **Figure 1**.

Before starting the programming portion of this assignment, complete and SUBMIT the following:

Q1. For each state, determine which event(s) cause a transition, and to which next state. Use the layout below to represent your answer. The entries of the matrix should indicate either a dummy effect, or the specific effect name(s) that is (are) executed when the transition is fired. Use the event names shown on the FSM diagram.

States	Accepting	Processing	Manufacturing	Shipping
Accepting				
Processing				
Manufacturing				
Shipping				

Q2. For each state, list the ***entry***, ***do***, and ***exit*** activities. Use the below layout.

State	Entry	Do	Exit
Accepting			
Processing			
Manufacturing			
Shipping			

Implementation of the Effects & Activities of the System

All activity and effect functions should only print out a short phrase simply announcing themselves. No real computations are done in this phase of the project.

Dispatching Events to the System

The events will all be entered by the user as a single character input according to the following encoding:

Character Input (all UPPER case)	Event
O	An order has been received
V	The payment method obtained from the client has been validated
I	The payment method obtained from the client was rejected (i.e. expired, or credit limit exceeded)
F	The factory line(s) failed to manufacture ALL requested replicas of the item being ordered
C	The factory line(s) successfully manufactured ALL requested replicas of the item being ordered
R	Delivery of the shipment to the client's address has been confirmed
L	The shipment was not delivered to the client (for some reason)
X	Terminate the program and exit immediately. In future phases, some clean up may be necessary

Your assignment:

- Answer questions **Q1** and **Q2** above.
- Implement the state diagram using the ***state pattern*** design methodology. The program must be written in ANSI C. The implementation must make use of pointers to functions.
- Each state should be implemented by a pair of `.h` and `.c` files, e.g., `accepting.c` and `accepting.h` that also includes the implementation of any `entry`, `do`, or `exit` activities.
- All effects on the transitions should be implemented in two files called `system.c` / `system.h`
- All entry, do, and exit actions of any state should be implemented in the corresponding state's `.c` file.
- In this phase of the project, the only input expected from the user is the character code of the event being dispatched. The system should continuously read the next event from the user until an 'X' for "exit" has been entered.
- The output of your program should clearly indicate which state the order is in, and what is being done for each user input. All transitions that occur should be evident from the output. Overall the status of the system should be clear.

A program that does not compile, or a program whose execution fails will result in a zero. All compilation and testing will be performed on `stu.cs.jmu.edu`, or the Linux computers in the lab; No exceptions. Assuming your implementation compiles and executes, the breakdown of the grading is given as follows:

- Answers to Q1 and Q2 5%
- Implementation using state pattern methodology (section 3.4) 95%

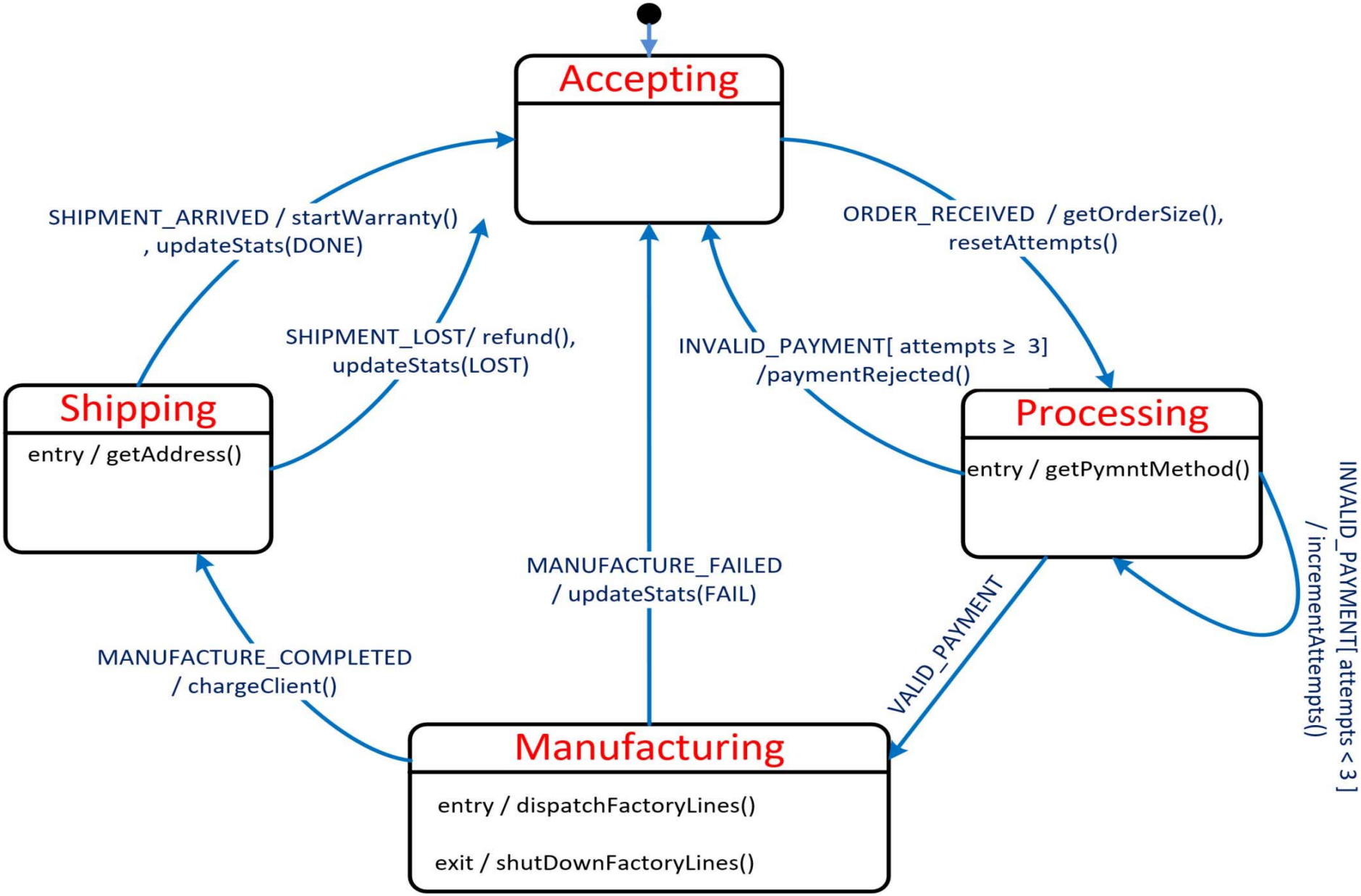

Figure 1: The *e-Build* Finite State Machine Diagram