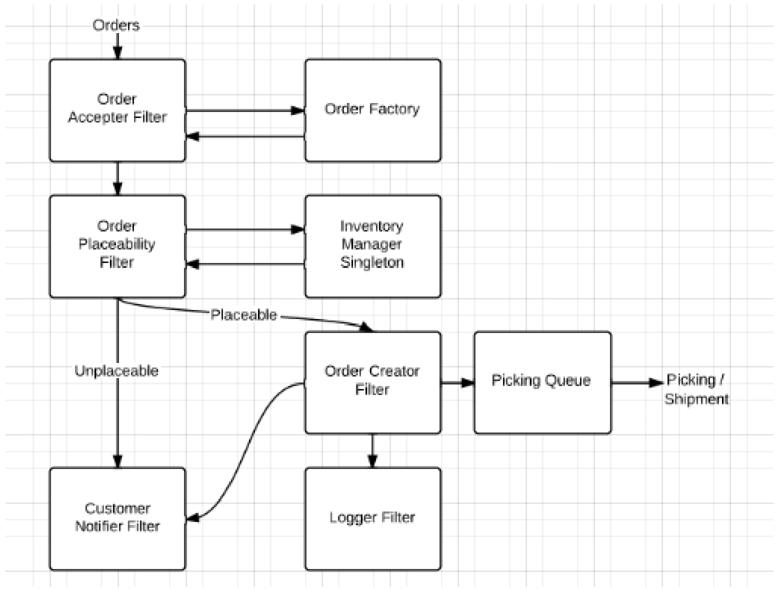
Samuel Henry
CSPP 51050 - Final Project Proposal

For my final project I will simulate order processing for a company that sells physical goods. This company accepts orders from customers, places orders if goods are in stock, logs transactions, emails order confirmations to customers, maintains inventory, picks goods from its warehouse(s), and ships the picked goods. I will implement everything up to warehousing/picking/shipping the goods. My design allows for many order input methods (e.g. web, email, fax, EDI, postal, in-store, etc) to flow into this order processing system. My design also allows the company to begin selling electronic goods by adding a new concrete order type and visitation implementation in its placeability and creator components.

For architectural patterns, I will use pipes and filters and a message queue. I will use pipes and filters to push orders through each stage of the process, encapsulating tasks into discrete stages that can pipe an order to each other. I will use a priority message queue to buffer picking messages so that available pickers can select work based on priority without the rest of the system needing to worry if the goods will be picked and shipped.

For design patterns, I will use Factory, Visitor, and Singleton (and the basic internal Iterator in Collections) to move orders through the system. My order Factory will create 1 day, 3 day, or 7 day orders based on delivery type and order parameters passed in from the Order Accepter Filter. This will allow implementation of filter-specific behavior not essential to an order to be performed through visitation. An order will be visited in an Order Placeability Filter. This visitation will determine order placability in terms of the concrete order type, the customer's distance from the warehouse/days in transit, and the inventory status for each line item. The Inventory Manager will be a Singleton (we only want one possible view of the company's inventory). It will restock items after an order is unplaceable due to a line item being out of stock (not the greatest strategy, could attach a stocking strategy to each SKU...). After checking placeability, an unplaceable order will be piped directly to the Customer Notifier Filter with a message to the customer. A placeable order will be piped to an Order Creator Filter. The Order Creator Filter will (a) visit a placeable order and pipe it to a Picking Queue (with priority based on delivery type), (b) pipe the order with a placed order message to the Customer Notifier Filter, and (c) pipe the order to the Logger Filter, which will record the transaction in the company's records. The Picking Queue will hold orders to be picked from stock based on priority.

Process Diagram:



Classes:

Order (Abstract -- customer id, line items/price, customer location coordinates)

Concrete Orders implement accept: for Visitors
OneDayOrder
ThreeDayOrder
SevenDayOrder

Pipe
Filter (Abstract)
OrderAccepterFilter
OrderPlaceabilityFilter
Customer NotifierFilter
OrderCreatorFilter
LoggerFilter

Visitor (Abstract)
PlaceabilityVisitor
PrepareForPickVisitor

PickQueue
PickMessage

Deliverable: I will submit two filed out .st source files: the implemented classes and the test cases.