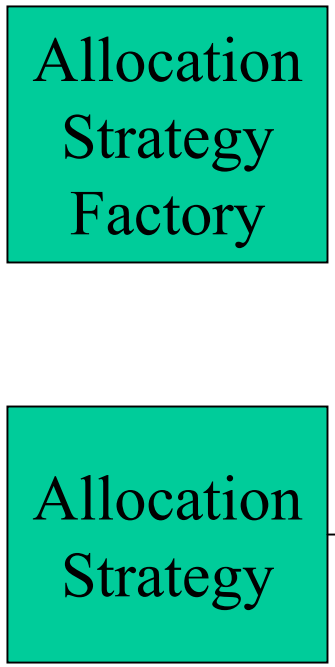
Accepting an Order

- OrderHandlingService.acceptOrder called with new order
- Order object create and saved in database
- BrokerService.acceptOrder called to continue processing
- BrokerService looks up Broker for ProductClass of order
- Broker.acceptOrder is called
- Broker calls BrokerStrategyFactory (implementation is configured in Broker.xml) to determine how order should be handled
- BrokerStrategy returned from factory is selected by type of order and state of product.
- BrokerStrategy.acceptOrder is called.
- Strategy for limit order when product is OPEN will create command to process order and put the command on a queue.

Processing an Order

- New order command is dequeued from command queue
- Command.execute is called which calls BrokerProcessor.processOrder
- BrokerStrategy.isActive is called to see if trading should be done
- OrderBook.crossesMarket is called to determine if order can trade with current market in order book.
- OrderBook.getPrioritySets is called to get orders and quotes from the book that can trade with the new order. Detail about allocation strategies are in the following slides.
- TradeService.tradeMatch is called to create matched parties from trade participants
- TradeReport.completeTrade is called to fill report all orders and quotes
- OrderBook.refreshBook is called to publish new current market. External call was done so that only one current market will be published even in complex transactions.

Allocation Strategies



Factory is used to get allocation strategies by TradingSession and ProductClass. Factory implementation is specified in OrderBook.xml.

Current elements:
- Price/time*
- Pro-rata*
- Customer
- DPM – fixed %
- DPM – var. %
- Market turner

* Must select one of these

Strategy is made of one or more elements that will each select items from the book.

- Allocation strategies are used to remove orders and quotes from the order book at the time of the trade.
- Configuration of allocation strategies is done by trading session and class in the TradingPropertyService.

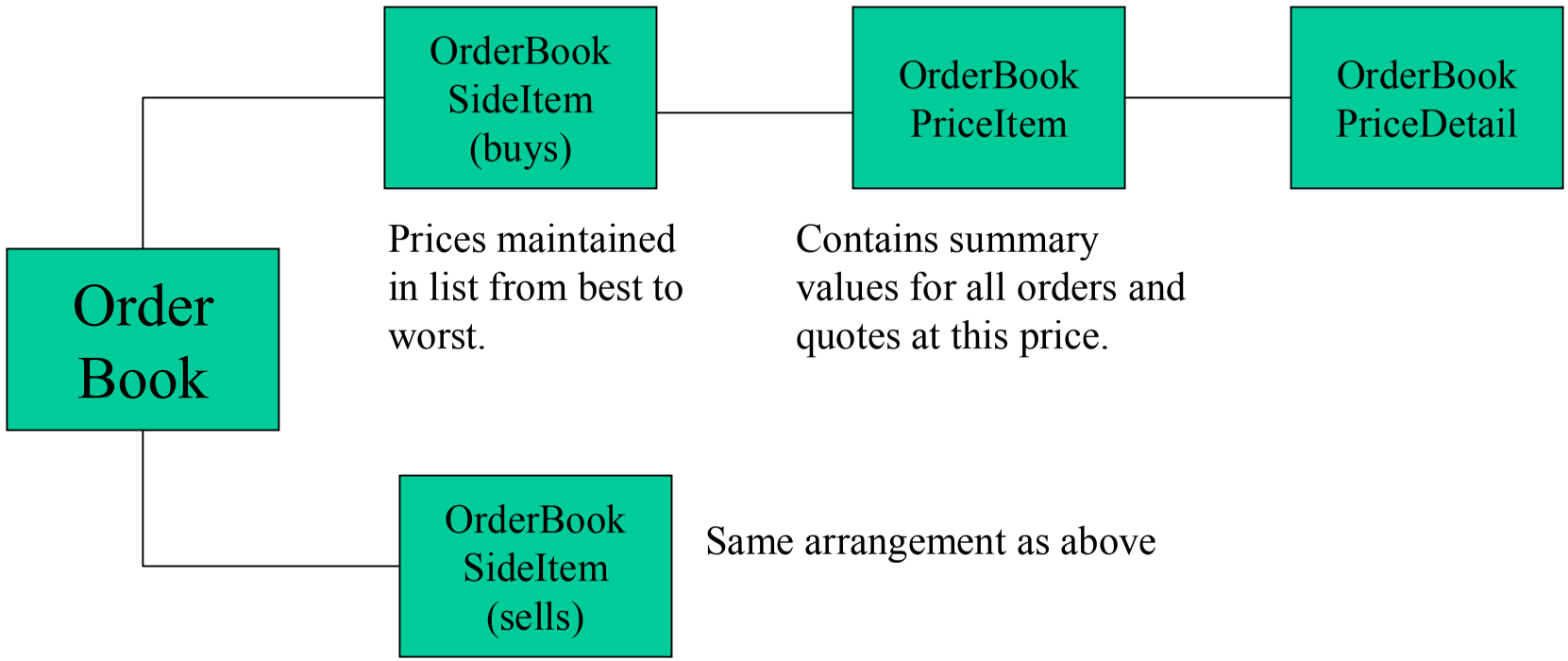
Allocation Example

- Current market for a product is 100@10.2 x 100@1.3, 30 of sell side is customer orders
- Buy order for 50@1.3 is received
- Allocation strategy is: Customer, DPM-Fixed%, Price/Time
- Total customer quantity on sell side is 30, so these are allocated to the trade
- DPM gets 30% of remaining 20, so 6 is allocated to DPM.
- Remaining quantity of 14 is allocated to orders and quotes in price/time order. DPM may participate in the trade a second time.
- AON/FOK orders are always tried last
- Only single pass is made with AON/FOK orders is tried, there is no logic for a best-fit or to try all combinations.

Allocation Restrictions

- Current processing of opening trade does not use allocation strategy. Orders and quotes at partially traded price are processed in price/time order.
- Contingency orders do not participate in an opening trade.
- Normal trading only takes place when product is OPEN.
- Trading of Strategy products is different than regular products, but allocation strategies will be used.
- If a trade is required when a Strategy product transitions from ON_HOLD to OPEN, the trade is similar to an opening trade of a regular product.
- The allocation of orders and quotes for a trade is different than matching trade records of reported trades since you would not normally only select partial quantity of a trade record.

Order Book


OrderBook SideItem (buys)
OrderBook PriceItem
OrderBook PriceDetail
Order Book
Prices maintained in list from best to worst.
Contains summary values for all orders and quotes at this price.
OrderBook SideItem (sells)
Same arrangement as above