Domain analysis

**Domain definition**:

The domain is “fashion inventory management”, the motivation of this domain analysis is to meliorate inventory management upon the existing systems and the addition of native QR code detection, with increase in on-line sales as an outgrowth, for the fashion retailer *WEARTHIS*.

**General knowledge, terminology and methodology about the domain**:

*Inventory management*

The supervision of non-capitalized assets (inventory) and stock items. Inventory management supervises the flow of goods from manufacturers to warehouses and from these facilities to point of sale. Simply, it’s about having the right inventory at the right quantity, in the right place, at the right time, and at the right cost.

Inventory management system

System that tracks inventory movement across all sales channels in real time.

*Inventory management technique*

*Just in time (JIT)*: The method lessen the volume of inventory that a business have. Only purchase inventory a few days before it is needed.

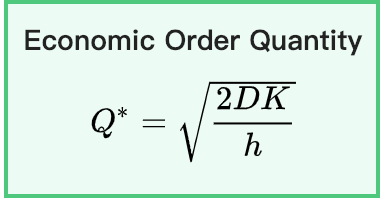
*ABC analysis*: The method based on the Pareto Principle, different inventories are prepared according to the price/value of the item.

*Dropshipping*: This method directly transfer customer orders and shipment details to manufacturer or wholesaler, who then ships the goods directly to customers.

*Economic Order Quantity*

The Economic Order Quantity (EOQ) is the number of units that a company should add to inventory with each order to minimize the total costs of inventory.

It is given by:

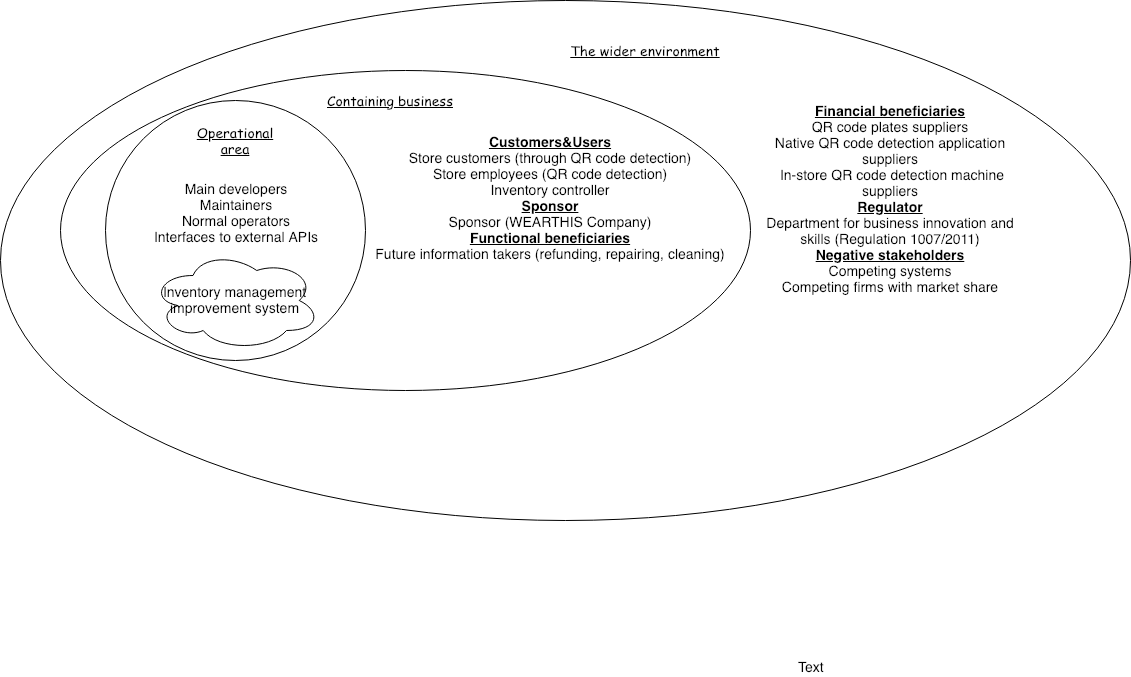


Where *D =* annual demand quantity, *K* = annual fixed cost (not per units), *h* = annual holding cost per unit.

*Backorder*

*Backorder* is an order for a good or service that cannot be filled at the current time due to a lack of available supply. *Backorder costs* include cost incurred by a business when it is unable to fill an order and must complete it later, usually computed and displayed on a per-unit basis.

**Customers, users and stakeholders**:



**APIs and procedures currently performed**:

*Existing APIs*

* *Inventory System API*:

Manages information about stock availability in different stores.

*Methods*:

1. *getAvailability(productCode: ProductCode, storeCode: StoreCode, size: int) : int* – returns the number of items of the specified product and size available in stock at the specified store.
2. *findClosestStores(addressIdentifier: String):Set* – returns a set of store codes for stores in the vicinity of the given address identifier. If the address identifier cannot be understood by the system or if there are no stores in a 50 mile radius, an empty set is returned.
3. *getStoreDescription(store: StoreCode): String* – obtains a human-readable, HTML-formatted description of the specified store.

* *Order System API*:

prepares and fulfils on-line orders.

*Methods*:

1. *createNewOrder(customerDetails: CustomerDetails):Order*

– creates a new order that can subsequently be filled with specific order elements using the other operations. CustomerDetails is a simple class that takes information such as name, address, etc.

1. *addItem(order: Order, product: ProductCode, size: int)*

– adds the specified item to the specified order.

1. *getOrderTotal(order: Order):Money*

– calculates the total price of the given order in a format that can be passed to the payment service.

1. *submitOrder(order: Order, paymentConfirmation: PaymentConfirmation) : boolean*

– triggers the delivery of the specified order given a secure payment confirmation code (which can be obtained from the payment service). Returns true if the order was successfully moved to fulfilment state. After this, you can no longer add items to the order.

* Payment System API

An external API that handles credit card payments.

*Methods*:

1. *processPayment(creditCardDetails: CardDetails, amount: Money) : PaymentConfirmation*

– processes the specified payment and returns a payment confirmation if successful (or null if the payment was declined for any reason). CardDetails is a class with field for the usual card information.

**Data analysis**:

The aim of this data analysis is to find statistical evidences which give an rough idea about some problems might be facing.

Maximum request rate approximation

This approximation use resource from real-world clothing shops statistics.

How it is done: Taken the annual sales / revenue within the UK market from three large clothing retailer --- H&M, Next, Newlook --- then dividing by factors according to assumptions made:

1. Assuming every order has price from 0 to 100 pounds and taking the average 50.
2. Assuming the daily working hours are 10 hours.
3. Assuming that all the sales / revenue come from in-store purchases.
4. Assuming our system are going to cover all of the in-store purchases as maximum.
5. Assuming that each purchasing process contains 10 requests to the CPU.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Retailers | Annual sales / revenue (Millions) | Annual working Second | Price per order | Maximum visiting rate (per second) | Maximum request rate (per second) |
| Next (2016) | 2374 | 365 \* 10 \*3600 = 13140000 | 50 | 4 | 40 |
| Next (2017) | 2305 | 13140000 | 50 | 4 | 40 |
| H&M (2016) | 15058 | 13140000 | 50 | 23 | 230 |
| H&M (2017) | 14580 | 13140000 | 50 | 22 | 230 |
| Newlook (2016) | 1491 | 13140000 | 50 | 2 | 20 |
| Newlook (2017) | 1455 | 13140000 | 50 | 2 | 20 |

(resources from:

https://www.nextplc.co.uk/~/media/Files/N/Next-PLC-V2/documents/2017/Copy%20of%20WEBSITE%20FINAL%20PDF.pdf

<https://about.hm.com/content/dam/hmgroup/groupsite/documents/en/Annual%20Report/Annual%20Report%202017.pdf>

<https://www.newlookgroup.com/system/files/uploads/financialdocs/fy17-annual-report-06-06-2017.pdf>)

As it shown, the maximum request rate (per second) will stay under a hundred and the safe position for our system will be around 200 request per second with necessary cascading failure prevention applied.

**Similar and competing software**:

1. Alipay technology
2. QR code WIFI connection