Michael Forbes
PFC
Edinburgh

Dear Mr Forbes

Many thanks for the opportunity to quote for PFC's sales-order processing system. As mentioned when we spoke, the first step is to define PFC's needs by analysing your current system and forming a 'problem definition' – a specific statement of what the computer system should do.

While much of this may seem to be a rehash of what you've said, it is vital to ensure we work on the correct problems.

Once this has been refined with you, we can then write software and install it on your computers. I hope the following is acceptable to you and I look forward to working further with you.

Yours sincerely

Bill Smith
Software Unlimited

1 Problem definition, system development methods and analysis of existing system

1.1 Problem definition

The basic problem is that PFC's departments all keep their own records, mostly on paper. This has the following 'symptoms':

- It is hard to tell which products sell best.
- Records of customers and orders, including notes about special treatment such as discount levels, are fractured between departments. Hence, it is hard to find out which invoices are outstanding for any customer or which orders have not been paid.
- It is difficult to tell how many units of any product are currently on order.
- It is difficult to find out how many of any particular product you have in stock.
- In summary, it is difficult to obtain the information needed to run your business. For example, knowing which products sell best and which are on order would allow more efficient control of raw materials, worker allocation and despatch. You wish to stock smaller numbers of expensive products and larger numbers of cheaper products, and to manufacture higher numbers of products that sell quickly. All of these are currently attempted ad-hoc, rather than being controlled.

The basic problems are to automate PFC's orders system and to create management information systems (MIS).

1.2 Software development methods

Traditionally, software has been developed using linear methods that follow prescribed stages such as:

1. plan (requirements analysis, followed by design)
2. build
3. test
4. deploy and maintain (adapted from Ganeshan, 2011)

The classic linear method is the waterfall method (Winston, 1970). Its main problem is that if the requirements are not fully understood, or there is an issue with the design stage, the final product will be unfit for purpose. It would be time-consuming and expensive to repeat these stages. (Strictly speaking, because it may be impossible to know all requirements in advance, it may be impossible to complete stage 1.)

Further, requirements may change as work progresses. For example, VAT rules might change. Also, software developers are not experts in other markets: they need customer insight during development to ensure software meets customer needs.

Attempts have been to improve on the waterfall model. (In fact, Winston's paper was a critique of the classic waterfall method.) One of these is the V-method (Waterfall Model, 2012). This adapts the waterfall model into two branches: design and validation. As each stage of the design branch occurs, appropriate testing is done, instead of the waterfall model's final testing stage. For example, individual software modules are unit-tested and the overall product is acceptance tested. However, this model still relies on requirements not changing. Also, acceptance testing is not done until the end. Hence it is still possible to deliver software that is unfit for purpose. There seems to be little scope for customer insight - customers might be involved with unit-testing but this may not give them a view of the complete product

The currently preferred method is *agile* ([Beck et al, 2001], [Phatak, 2010]). In this method, after an initial requirements analysis, development is iterated, concentrating on quickly creating working software which meets customers' 'must do' requirements. Software is regularly tested by customers as development continues: this can allow customers to discover further requirements that can be satisfied in subsequent iterations. With customer agreement, 'should do' and 'nice to have' requirements can be postponed or dropped if the project runs out of time or budget.

Customers become part of development teams, influencing functional and non-functional requirements. We suggest that either you or your sales director, along with some Orders and Despatch staff, join this project's team. This may allow greater insight as different parts of PFC become more familiar with each other's needs.

As agile methods have developed, so too has object-oriented programming. Objects are individual items associated with a system. For example, each individual customer record is an object. (The set of all customer records is the customer record *class*.) All objects in a class have the same set of attributes, e.g. name, address, status, but may have very different values for these attributes. There can be an indefinite number of objects in each class, so that there is no limit to the number of customers, or the number of lines in an order. Also, all objects in a class have the same capabilities. For example, a customer's phone number can be changed but an invoice cannot even have a phone number. The 'internals' of objects are hidden from the rest of the system - objects respond to messages by changing their attributes and/or providing information about themselves. Object orientation fits well with agile methods because objects can be prototyped with 'must-do' requirements in early iterations, then tested in action by customers. Other requirements can be added in later iterations.

1.3 Proposal for new system, requirements analysis and analysis of existing system

1.3.1 Proposal for new system

The new system should not change your business methods but should improve on them. For example, you currently process returns in certain ways. The system should not require these to change but should enable accurate returns processing. The system should also give you the management information you require. Hence MIS functional requirements are among our *assumptions and suggestions*. Please advise whether these are correct/suitable.

To avoid repetition, analysis of the existing system (including <u>current practices</u>) is included in the following requirements tables. Detailed proposals are included in the functional requirements, use case descriptions and activity diagrams.

1.3.2 Requirements analysis

This is a list of the things the new system must or should do (functional requirements) along with the qualities the new system should have (non-functional requirements). You will see that many of the requirements simply automate your current practices.

1.3.2.1 Functional requirements

The system must automate orders from initial approach by a customer, via formal order, picking and despatch, invoicing to final payment. The list of requirements is:

1.3.2.1.1 Products

Manufacture enters details, apart from price, for subsequent new products. We are happy to enter details of existing products if you supply them.

FR1	Hold products' details (PFC currently has around 120 products.)
FR2	Add new products
FR3	Each product has a set price. Only Mr Forbes or the Sales Director (SD) can set prices.
FR4	Each product has a code generated from its room, type, material and finish. For example, white-painted teak kitchen chairs have product code KI-CH-TK-P-w.

1.3.2.1.2 Customers

FR5	Hold customers' details
FR6	Add new customers.
FR7	Sales staff (SS) can **suggest** discount rates but only Mr Forbes/SD can set rates. (Rates vary between customer: once a rate is set, it is applied to the total value of each subsequent order from this customer.)
FR8	If a discount rate has been suggested, *the system notifies Mr Forbes/SD* and puts that customer on hold until Mr Forbes/SD sets the rate.
FR9	Mr Forbes/SD can 'favour' and 'unfavour' customers. Favoured customers' orders are prioritised when stocks are low.
FR10	Mr Forbes/SD can 'hold' and 'unhold' customers (e.g. if a customer is believed to be a bad risk) as if they had failed credit-check.

1.3.2.1.3 Orders

Orders for held customers are entered into the system but are not picked until the customers are 'unheld'.

FR11	Enable receipt of orders by email
FR12	Process orders: Orders can be input by Orders staff only. (Orders can reach PFC by phone, fax, letter, *email* from customers or SS, or be placed by visiting customers.)
FR13	New customers who are organisations are held while they are credit-checked.
FR14	If an organisation fails credit-check, its orders go no further.
FR15	Organisations that have passed credit-check are allowed 30 days' credit from despatch of goods.
FR16	Orders from individuals must be accompanied by cheque payment – such orders are held until cheques have cleared.
FR17	If customers have bad debt (i.e. invoices unpaid more than 30 days after despatch), they are held until it is settled.

1.3.2.1.4 Picking and despatch

FR18	Create picking lists from orders
FR19	Process picking lists: • *All customers are content to accept part-deliveries.* • As orders are assembled, Despatch enters numbers picked of each item.
FR20	Create despatch notes

1.3.2.1.5 Invoicing and returns

FR21	• 4 working days after despatch, print *or email* invoices that take into account under-deliveries, over-deliveries and returns. • Final invoices are *electronically* supplied to Accounts.
FR22	Invoices may include delivery charges: delivery charge bands are dependent on distance from PFC. (You have supplied these bands.)
FR23	Authorise returns: • Returns must be notified within 3 working days of despatch. • Unauthorised returns. They are redelivered to customers. PFC invoices for redelivery. • Returns due to over-delivery and to goods being delivered damaged are accepted automatically.

FR24	Process returns: • If returns due to over-delivery are returned damaged, having been collected by PFC, Despatch manager is notified and customers are not invoiced. • If returns due to over-delivery are returned damaged via other means, Despatch notifies Orders, who invoice accordingly.
FR25	Process non-materialised returns: • If authorised returns are not received within 10 working days, customer are invoiced. These invoices will not include delivery charges.

1.3.2.1.6 Payments

FR26	Process payments • allocating them appropriately to unpaid invoices. PFC does not *and will not* accept cash payments. PFC accepts BACS payments. • PFC issues receipts for all payments.

1.3.2.1.7 Management information

The following reports can be run at any time. They are are automatically run and emailed to Mr Forbes/SD at the end of each month. The automatic ones cover the last calendar month and the last calendar year. These reports cannot be run by anyone apart from Mr Forbes/SD and the head of Accounts (FR28 only). The Despatch manager can also run a report on mis-deliveries and despatch staff involved with them (1st part of FR31).

FR27	*Provide analyses of customers in any required reporting period:* • *Those that have not ordered.* • *For each customer who has ordered, summaries of orders and invoiced goods, along with values.* • *Those that are currently and have been **favoured**.* • *Those that are currently and those that been **held**.* • *Those that currently have and those that have had **bad debt**.* • *List and summarise any customer's orders and purchases: value, customer status (favoured, held, normal), items ordered/bought, returns, including unauthorised returns*
FR28	*Provide bad debt analyses:* • *Customer, amount owed, age of bad debt and total amount owed by all bad debtors where bad debt is 30 or fewer days old.* • *As above but for bad debt aged 31 to 60 days.* • *As above but for bad debt aged 61 to 90 days.* • *As above but for bad debt aged more than 90 days.* *For each customer, list of all bad debts and suggested action (first warning, court action, etc) based on customer's hold or favoured status, total bad debt and actions history.* (Only Mr Forbes/SD can authorise actions.*) As they are taken, they are recorded on the system by relevant staff.*)

FR29	*Provide sales analysis:*
	• *For each product and product group (room, type, material, finish):*
	  • *Number sold*
	  • *Total and per-unit revenue*
	  • *Number, value and reasons for returns*

FR30	*Provide orders analysis*
	• *For each product and product group*
	  • *How many of each are on order, separating out those that are on back order, so that over- and under-manufacture are visible*
	  • *Revenues that would be made from these items*
	• *Despatched orders not yet invoiced*
	• *Invoiced orders not yet paid*
	• *Paid orders*

FR31	*Provide despatch and returns analysis* • *Mis-deliveries, including despatch staff involved* • *Unauthorised returns details* • *Returns due to damage and over-delivery (including returns due to over-delivery returned damaged)*

1.3.2.1.8 Customer browsing facility

FR32	The system must include a browsing facility in the reception area, where customers can look through an online catalogue. *This catalogue shows only pictures of products, their prices, descriptions and product codes. It is filterable by type (e.g. chairs, tables, etc), room (e.g. kitchen), material and finish. Customers are unable to use the facility to do other computing.*

1.3.2.2 Non-functional requirements

The system should:

NFR1	*Be easy to use:* • *Simple, robust log-in, requiring regular password changes* • *Colours and formats that suit colour-blindness and other visual impairment.*
NFR2	*Be secure:* • *staff can only see and alter data, and run reports, relevant to their roles* • *no unauthorised persons can use the system, apart from visitors using the browsing facility*
NFR3	*Be robust:* • *not crash easily* • *recover from crashes with minimal data-loss*
NFR4	*Have robust, automatic on-site and off-site back-up* *We suggest* • *a GFS (Sonora Corporation, 2007) tape system for on-site back-up* • *Crashplan Pro (Crashplan, 2012) for off-site backup.*

2 Use cases

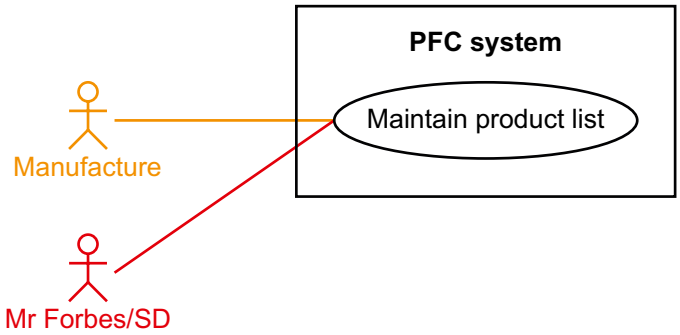
2.1 Use case diagrams

These show who is involved with each basic business process. <<include>> shows that one use case always involves another. For example, despatching an order always includes creating a delivery note. <<extend>> shows that one use case sometimes includes another. For example, when entering an order, the order would be held and no picking list would be generated if the customer was on hold.

2.2 Use case descriptions

These give further detail about each use case, including the functional requirements associated with each use case.

2.2.1 Maintain product list



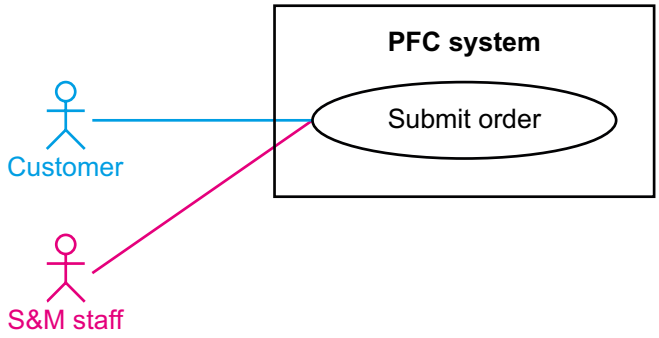
Actors: Manufacture, Mr Forbes/SD

Goal: to create and maintain a list of products

The product list contains details of all products, including discontinued products. (These do not appear in the browsing facility.) Each product has

- a price, set by Mr Forbes/SD. (Prices are suggested by Manufacture.) Until a product's price has been set, the price appears as '£TBC' in the browsing facility
- a room (e.g. kitchen)
- a type (e.g. chair)
- a material (e.g. teak)
- a finish (e.g. white painted)
- a code, generated automatically from its room, type, material and finish
- a photo
- a description (Descriptions are suggested by Manufacture but do not appear until finalised by Mr Forbes/SD.)

2.2.2 Submit order

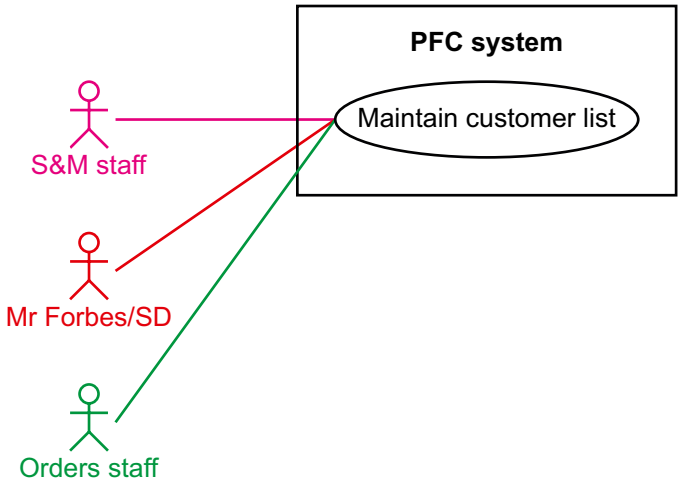


Actors: Customers, S&M staff

Goal: to submit orders to PFC Orders staff

A customer decides the PFC products he or she wishes to buy. Either the customer or a relevant S&M staff-member submits this order, by phone, *email*, fax or letter to PFC's Orders staff. (*S&M staff have smartphones and/or laptops. We will implement email for Orders staff using your existing copies of Microsoft Outlook.*)

2.2.3 Maintain customer list

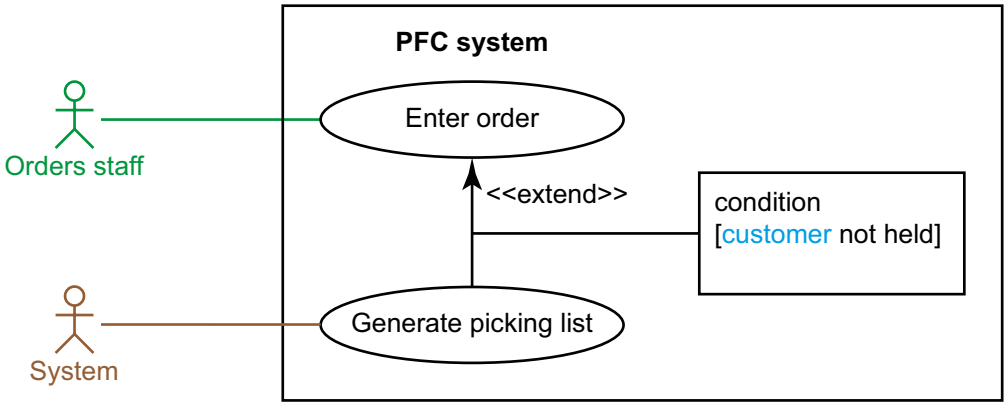


Actors: S&M staff, Mr Forbes/SD, Orders staff

Goal: to maintain up-to-date sets of customer details, so that customers' histories can tracked and products despatched efficiently

Either S&M or Orders staff enter or amend customer details, checking incoming orders for changed details and amending records appropriately. If a customer has requested a new or greater discount rate, S&M or Orders staff enter this: the system automatically emails Mr Forbes/SD and places the customer on hold until the rate has been set. Mr Forbes/SD can hold customers (e.g. those believed to be bad risks but who do not yet have bad debt). Mr Forbes/SD can also mark customers as 'favoured'.

2.2.4 Enter order and generate picking list

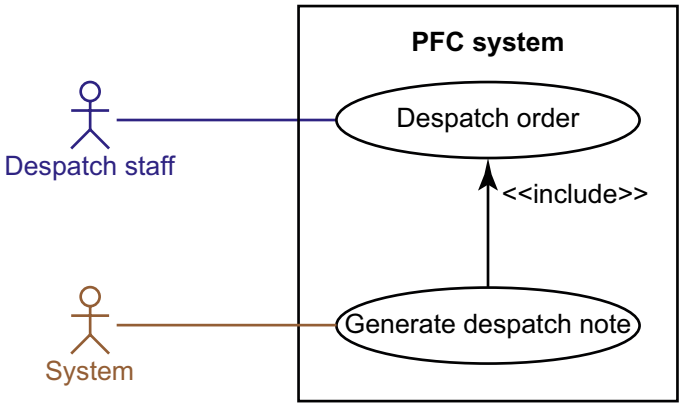


Actors: Orders staff, System

Goal: to get orders into system and ready for despatch

Orders staff enter orders into system. The system automatically checks whether customers have bad debt. If so, Mr Forbes/SD and Accounts staff are emailed and these customers' undespatched orders are held. (*Mr Forbes/SD can over-ride this hold for favoured customers.*) Orders not held due to bad debt or awaiting discount decision are forwarded to Despatch in the form of on-screen picking lists.

2.2.5 Despatch order

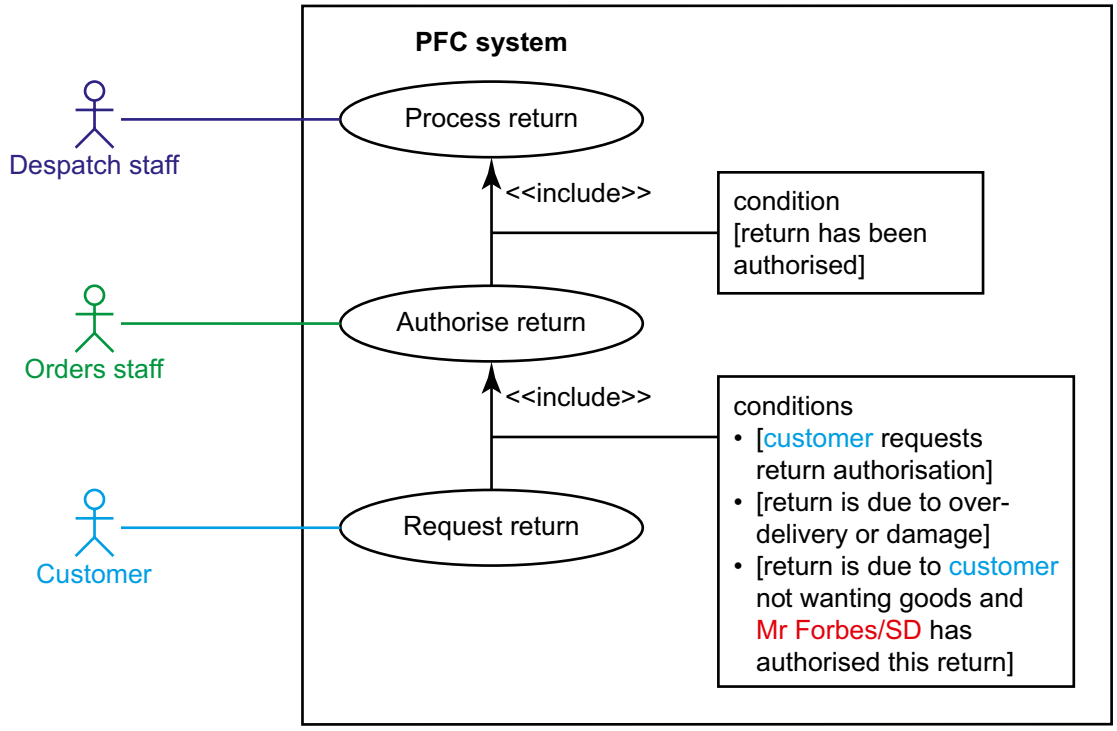


Actors: Despatch staff, System

Goal: to despatch goods to customers

Despatch staff assemble deliveries according to picking lists. As each order line is picked, it is 'ticked off' on screen. Sets of lines that have not been despatched are transferred to Despatch's 'priority to-do list' so that incomplete orders are completed as soon as possible. (*A daily or weekly summary of outstanding lines is forwarded to Manufacture.*) The picked set of products is then loaded onto a truck and printed as a despatch note. (*Discontinued products are noted on despatch notes.*) The despatched items automatically become an initial invoice. (See section 3.3.3.)

2.2.6 Process return

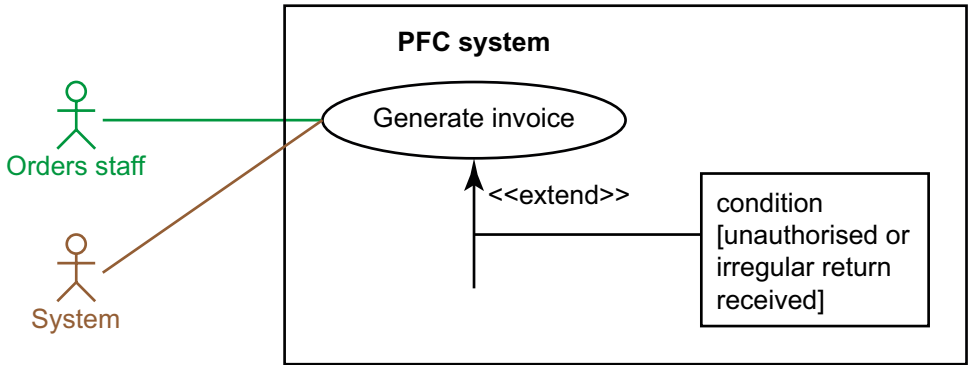


Actors: Despatch staff, Orders staff, Customer, (Mr Forbes/SD)

Goal: to receive expected returns

- If customers notify over-delivery or damaged goods within *3 working days* of delivery, Orders staff issue returns authorisations to both them and Despatch, noting claims of damage on authorisations.
- If customers wish to return undamaged, correctly supplied goods, this must be authorised by Mr Forbes/SD. PFC charges for collecting such returns.
- When returns are received, Despatch checks for damage, returning undamaged goods to stock and notifying Orders of receipt and discrepancies from authorisations. See Returns-processing activity diagram (section 3.3.2) for fine details of how discrepancies are handled.
- Unauthorised returns are not accepted: they are returned to customers, who are then invoiced for redelivery.

2.2.7 Generate invoice

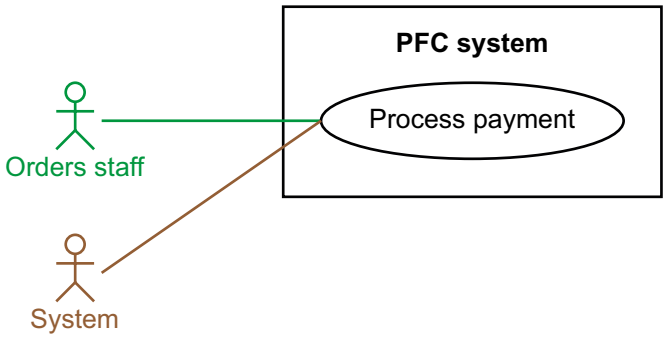


Actors: Orders staff, System

Goal: to bill customers appropriately

- *4 working days after despatch*, the system generates invoices based on despatch notes but taking into account authorised returns. (See section 3.3.3.)
- Orders staff create invoices for redelivering unauthorised returns and for returns due to over-delivery where the goods have been returned damaged (and so cannot go back into stock) and this is presumed to be the fault of the customer or its courier.

2.2.8 Process payment

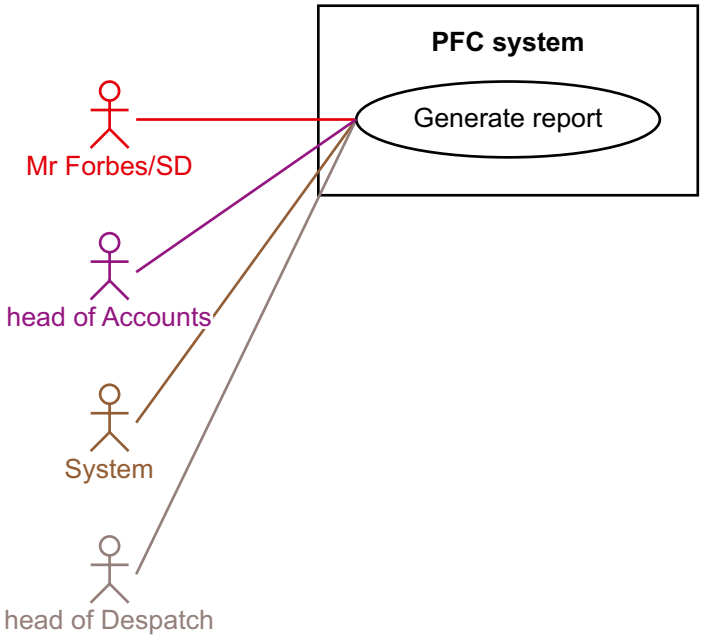


Actors: Orders staff, System

Goal: to deal with payments

- As cheque payments are received, orders staff allocate them to individual customers' undespatched orders or other customers' invoices for despatched goods according to the accompanying documentation. (If relevent data is not received, payments are allocated to the oldest unpaid invoice(s) for that customer. If this results in an invoice being part-paid, the system handles this, adjusting records of bad debt accordingly.) Orders staff then pass cheques to Accounts for paying into PFC's bank, keeping a record of cheques to cover undespatched orders. (Accounts provides a weekly list of such cheques that have cleared, so that relevant orders can then be despatched.)
- The system handles BACS payments appropriately, unholding customers/orders as appropriate.
- The system provides *daily/weekly/monthly* sets of information to Accounts.

2.2.9 Generate management report



Actors: Mr Forbes/SD, head of Accounts, System, head of Despatch

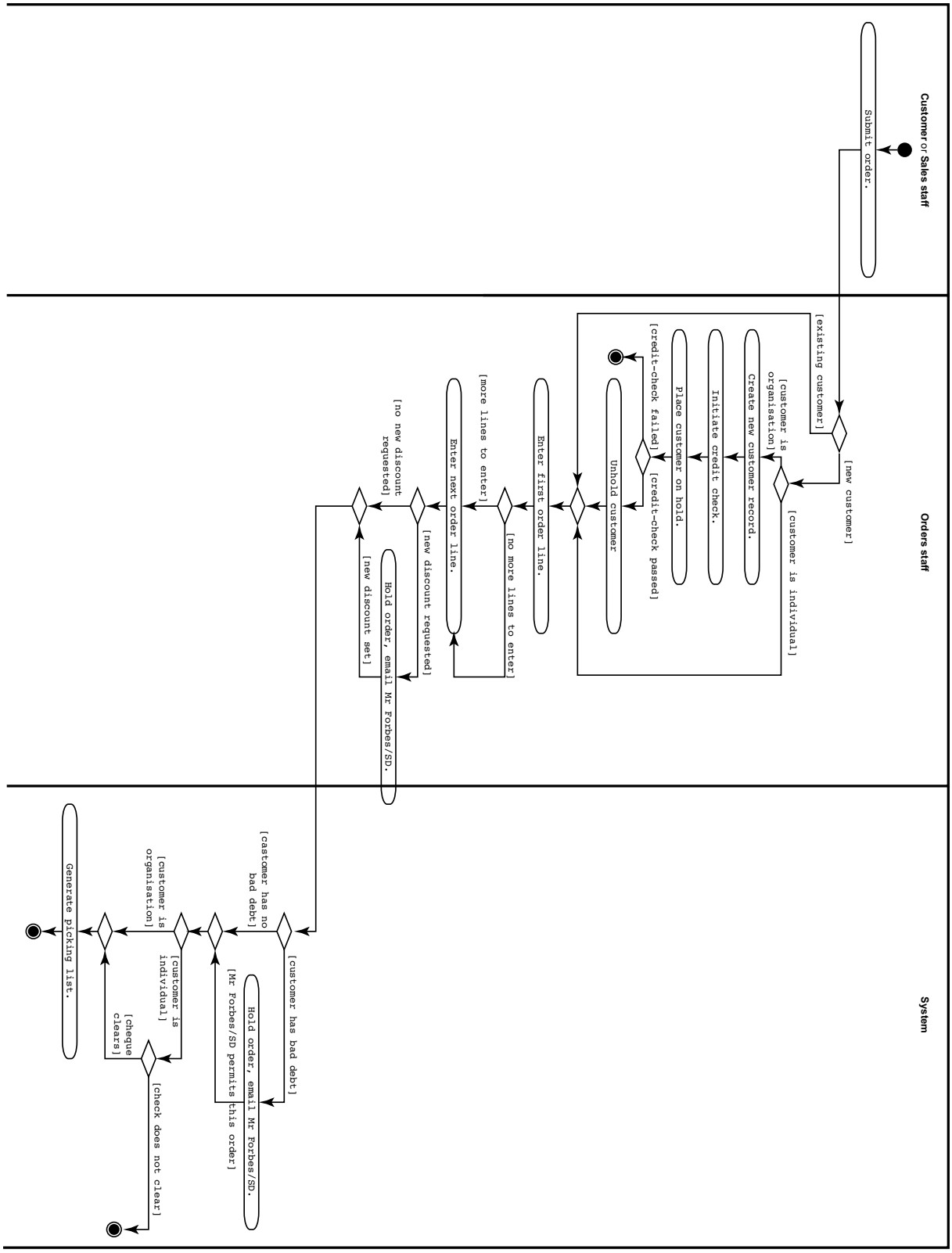
Goal: to provide management information and an overview of business

- Each month, the system generates a full set of reports for the past month and year, emailing them to Mr Forbes/SD and head of Accounts as indicated in section 1.3.2.1.7.
- Mr Forbes/SD and the head of Accounts can also run these reports at any time, for any reporting period they require.

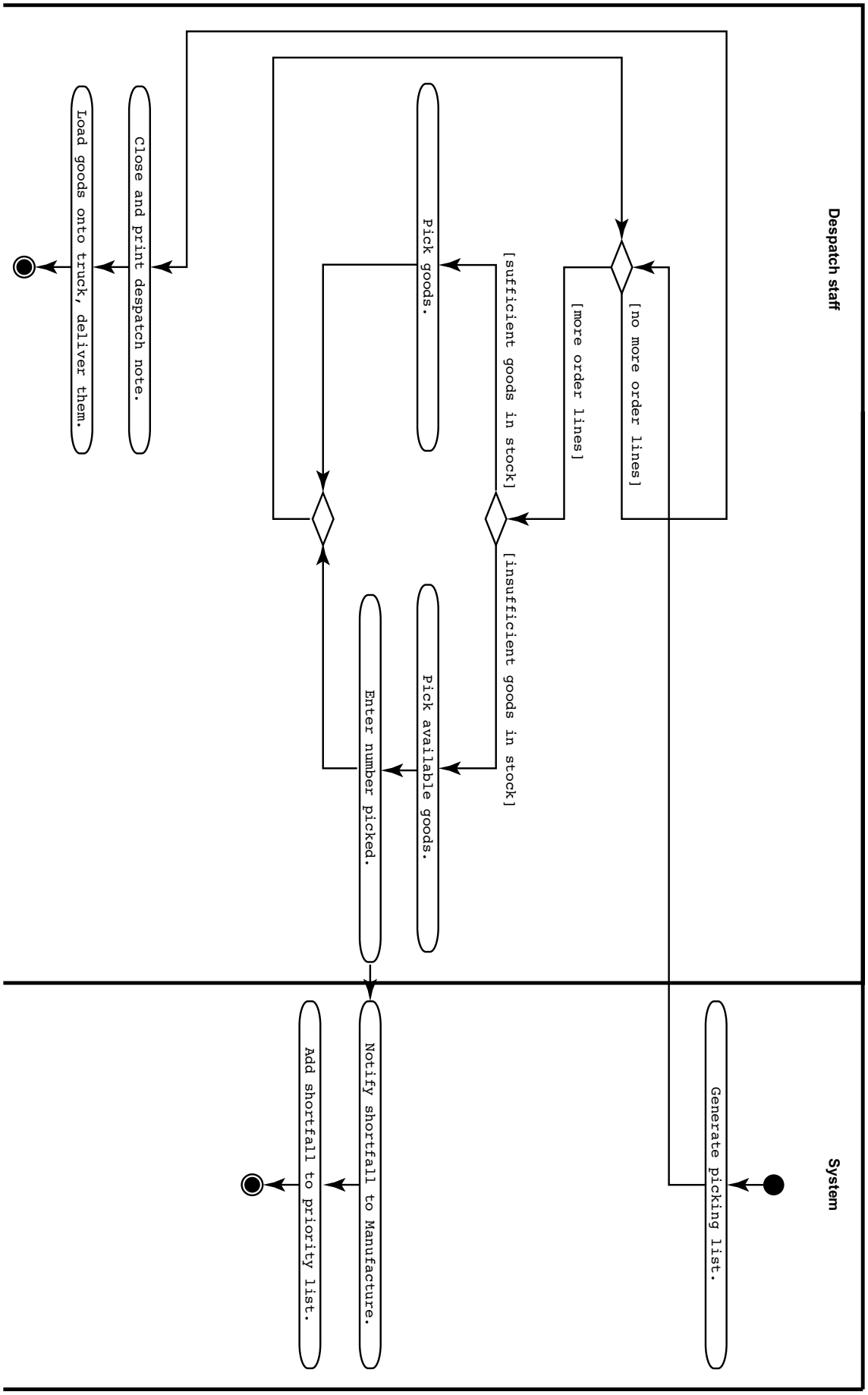
3 Activity diagrams

These show the processes and decision trees for the order process. ‘Swimlanes’ show who does what.

3.1 Submit and enter orders

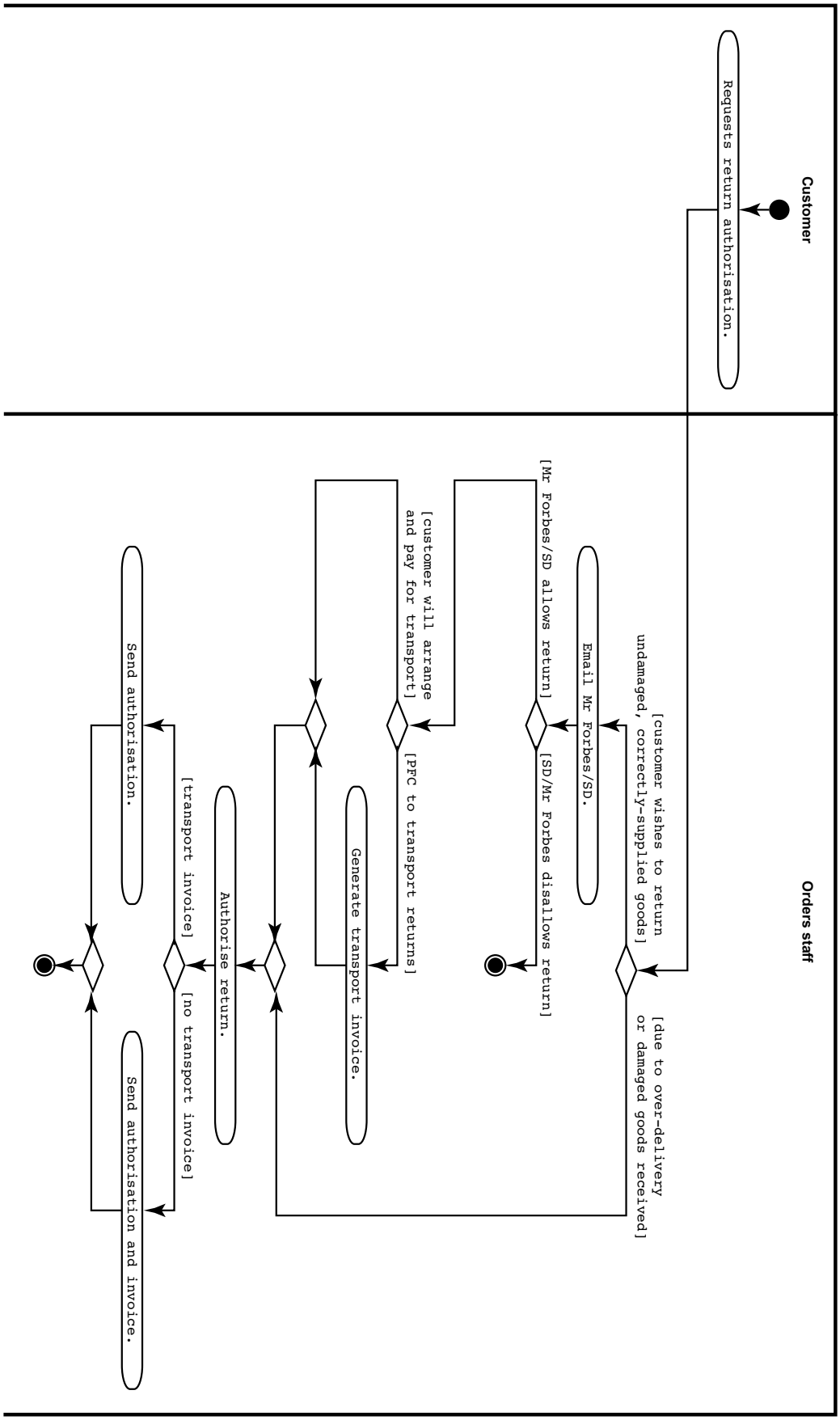


3.2 Despatch orders

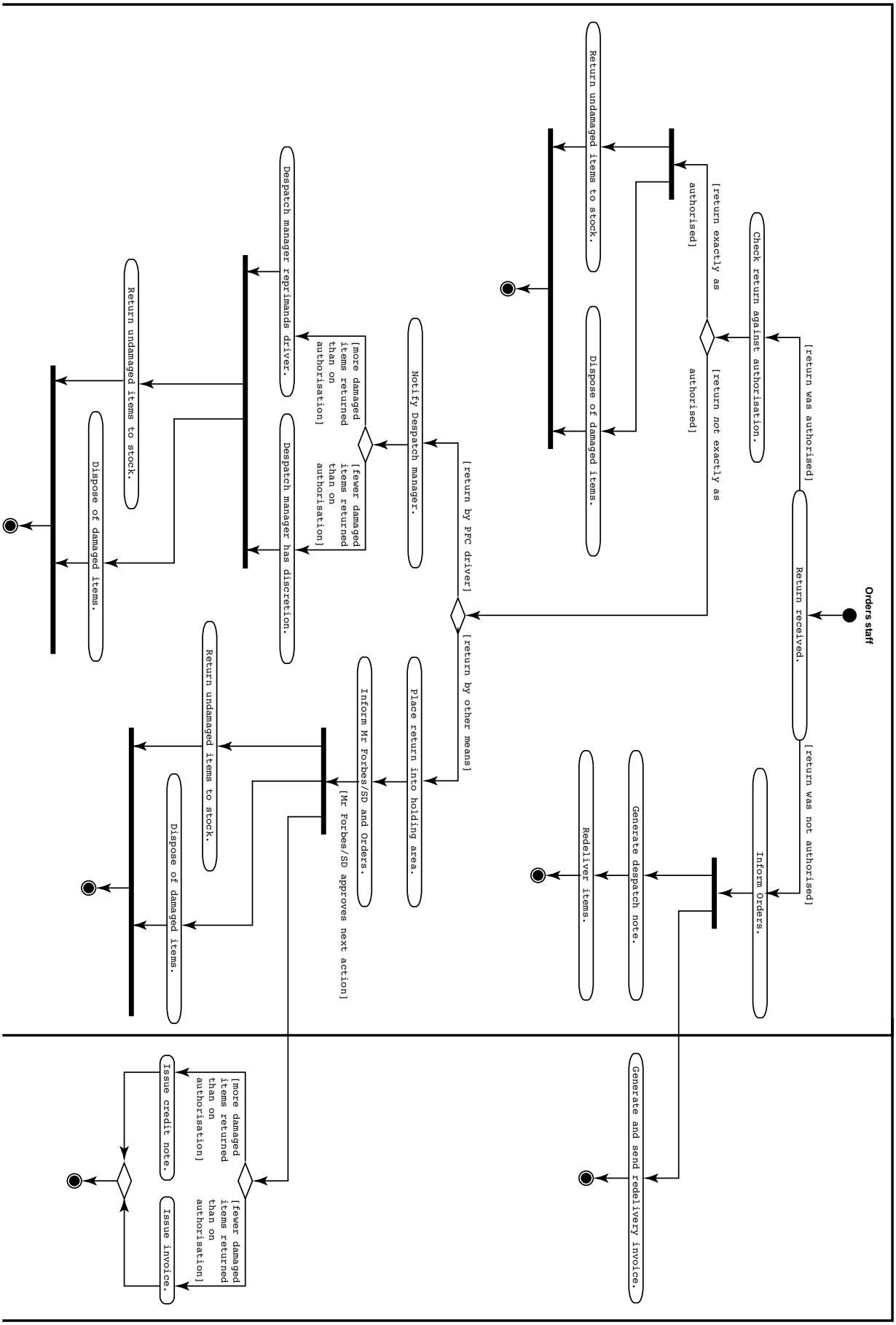


3.3 Returns

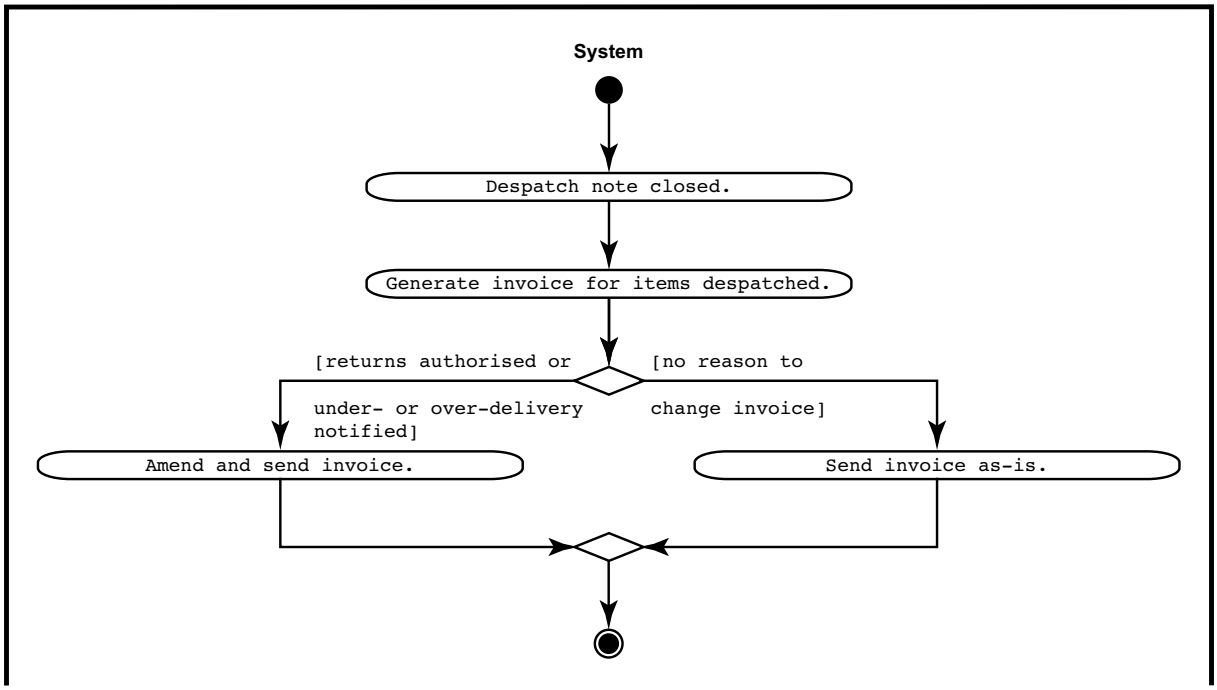
3.3.1 Returns authorisation



3.3.2 Process returns



3.3.3 Invoice customers for normal deliveries



4 Objects

The initial classes we suggest are:

4.1 Customer

Because PFC deals with many customers, this is the ideal place to start. Each customer has a name and address. Customers may also have contact names, phone numbers, discount rates, statuses etc. These are represented as

Customer
customerID name address landTel mobileTel fax email deliveryCharge contactName individualOrOrganisation suggestedDiscountRate confirmedDiscountRate newOrExisting heldOrNormalOrFavoured sortCode accountNumber usesBACS
findCustomer(name) recordDetails() editDetails() placeOrder() makePayment() suggestDiscountRate() confirmDiscountRate() makeExisting() setHeld_Normal_Favoured() setBankDetails Customer()

4.2 Products

Each product is modelled as follows. Product codes are automatically generated from products' rooms, types, materials and finishes. A product object is automatically 'current' when it is created but can be set to 'discontinued'.

Product
productCode productName room type material finish suggestedUnitPrice actualUnitPrice descriptionForCatalogue photo currentOrDiscontinued
generateProductCode() suggestUnitPrice() setOrAmendUnitprice() setDiscontinued() Product()

4.3 Orders

Orders are central to the system: picking lists are views of orders which do not show prices. In turn, despatch notes and invoices originate from picking lists.

Order
orderID date held completeOrIncomplete totalValue
setDate() setHeld() setUnheld() setComplete() setIncomplete() calculateTotal() Order()

4.4 Despatch Notes

Despatch notes are created from, and hence pull data from, Orders.

DespatchNote
DespatchNoteID date openOrClosed
setDate() SetClosed() DespatchNote()

4.5 Invoice

Invoices are created from closed despatch notes.

Invoice
InvoiceID date invoiceTotal
setDate() calculateTotal() Invoice()

4.6 ItemLines

Each order, picking list, despatch note and invoice contains one or more item lines. There are three types of item line, depending on whether they are in orders and picking lists, despatch notes or invoices.

ItemLineForOrder
productCode productName numberOfUnitsOrdered unitPrice priceForLine
enterProductCode() enterNumberOfUnits() calculatePriceForLine() ItemLineForOrder()

ItemLineForDespatchNote
productCode productName numberOfUnitsDespatched
changeNumberOfUnits() ItemLineForDespatchNote()

ItemLineForInvoice
productCode productName numberOfUnitsDespatched unitPrice priceForLine
changeNumberOfUnits() ItemLineForInvoice()

NumberOfUnitsDespatched is initially be taken from ItemLineForDespatchNote but can be amended in case of returns.

4.7 Payments

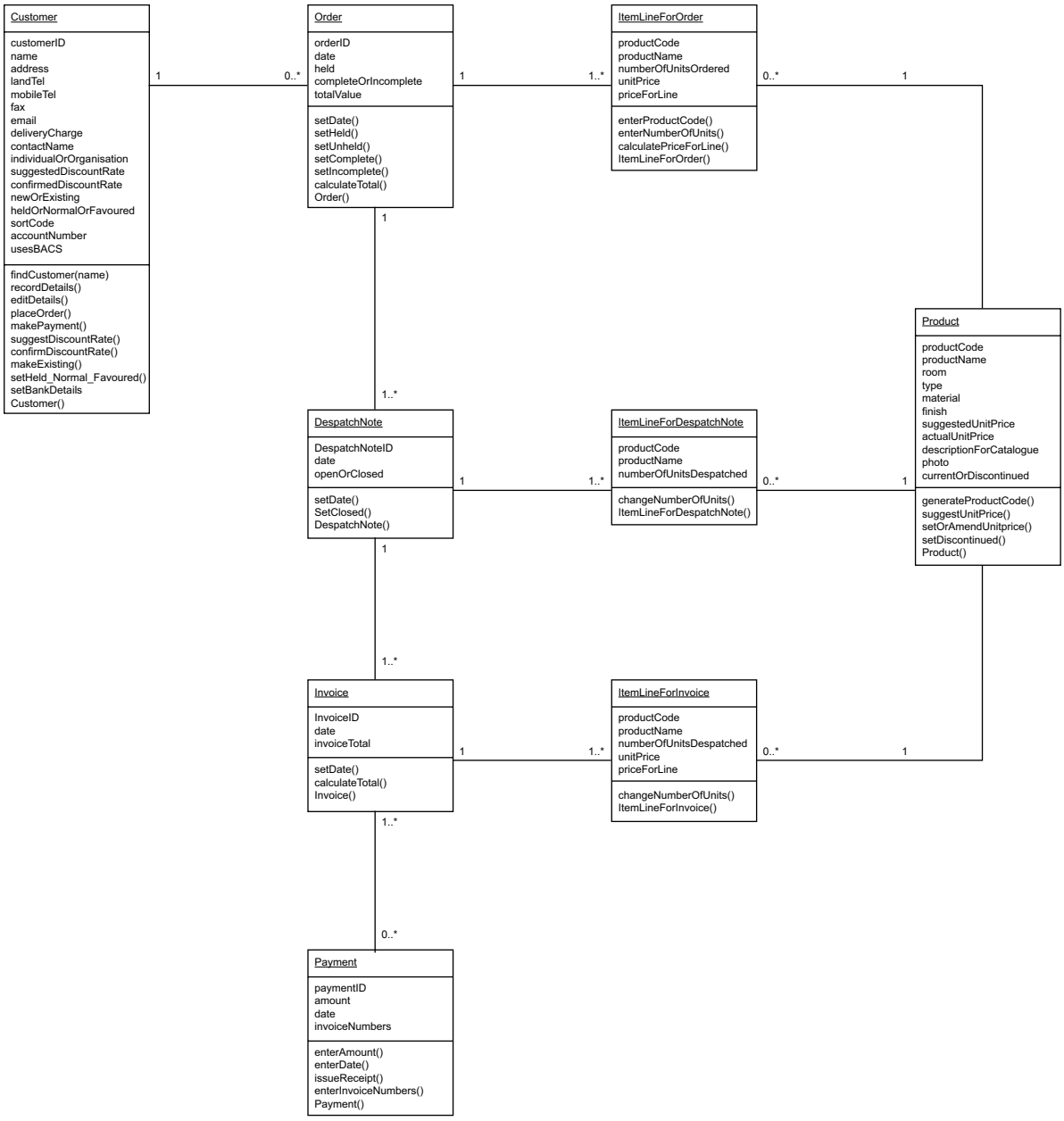
Payments are associated with one or more invoices. If the customer does not make this association, Orders staff do so.

Payment
paymentID amount date invoiceNumbers
enterAmount() enterDate() issueReceipt() enterInvoiceNumbers() Payment()

There will be more classes for reports, forms, etc but these do not need to be considered at this stage.

4.8 Class diagram

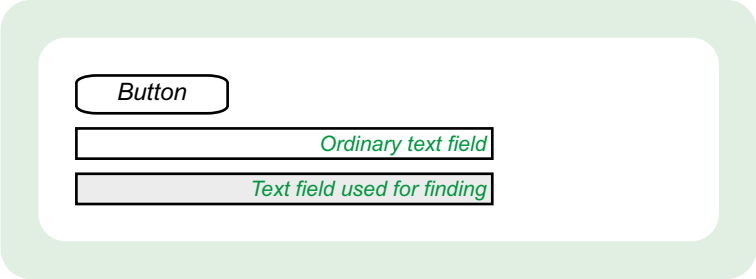
This shows how the classes are associated in the system. The links between the classes show, for example, that any customer may have placed 0 or more orders, but each order belongs to exactly 1 customer.



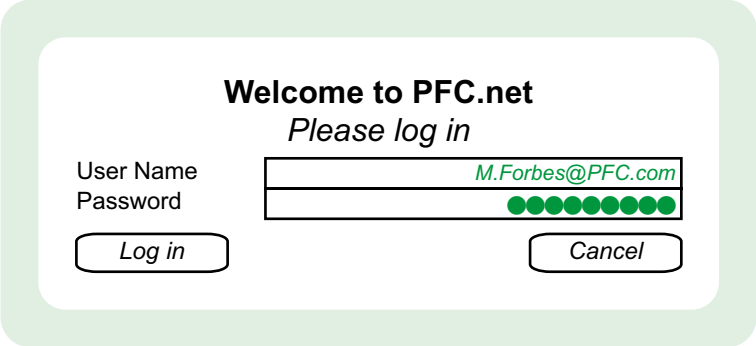
5 Storyboard

This section shows suggestions for the main screens staff and customers would use. Buttons and fields that staff may not use are inactive or invisible to them - this is controlled by their login IDs. For example, S&M staff can enter suggested discount rates and see approved rates but only Mr Forbes/SD can enter approved rates.

5.1 Key



5.2 Log-in screen



5.3 Create/find/amend customer

Create/amend/find customer(s)

Set mode

Find | *Amend* | *Find*

In *Find* mode, enter text into gray fields.

Customer Name	*Rob Kemmer*
Customer address line 1	*Merchiston Campus*
Customer address line 2	*Edinburgh*
Customer postcode	*EH10 5DT*
Customer contact name	*<AS CUSTOMER NAME>*
Customer full address	*Rob Kemmer* *Merchiston Campus* *Edinburgh* *EH10 5DT*
Phone	*+44 (0)131 455 2737*
Mobile	*+44 (0)7909 455 2737*
Fax	*+44 (0)131 455 2651*
Email	*rob@kemmer.com*

STATUS		*NEW*
		Change to EXISTING
		HELD
	Hold	*Unhold*
	Favour	*Unfavour*
		INDIVIDUAL
	Change to COMPANY	Change to INDIVIDUAL

DISCOUNT	
Suggested rate	*NOT YET SUGGESTED*
Approved rate	*NOT YET APPROVED*

BANK DETAILS	
Sort code	*08-92-85*
Account number	*13488565*
Uses BACS	*YES*
Change to NO	Change to YES

5.4 Create/find/amend product

Create/amend/find products(s)

Set mode

Find | *Amend* | *Find*

In *Find* mode, enter text into gray fields.

Product name	*Billy*
Product room	*Kitchen*
Product type	*Chair*
Product material	*Teak*
Product finish	*Paint (white)*
Product code	*KI-CH-TK-P-w*

PRICE

Suggested price	*£29·99*
Approved price	*£19·99*

STATUS *CURRENT*

Change to DISCONTINUED

DESCRIPTION *SUGGESTED*

Change to APPROVED | Change to SUGGESTED

A beautiful 4-legged kitchen stool, expertly made in hard-wearing, ethically-sourced teak, painted with white stain-resistant, low-VOC Farrow & Ball 'Pointing 2003' to fit with modernist kitchen décor.

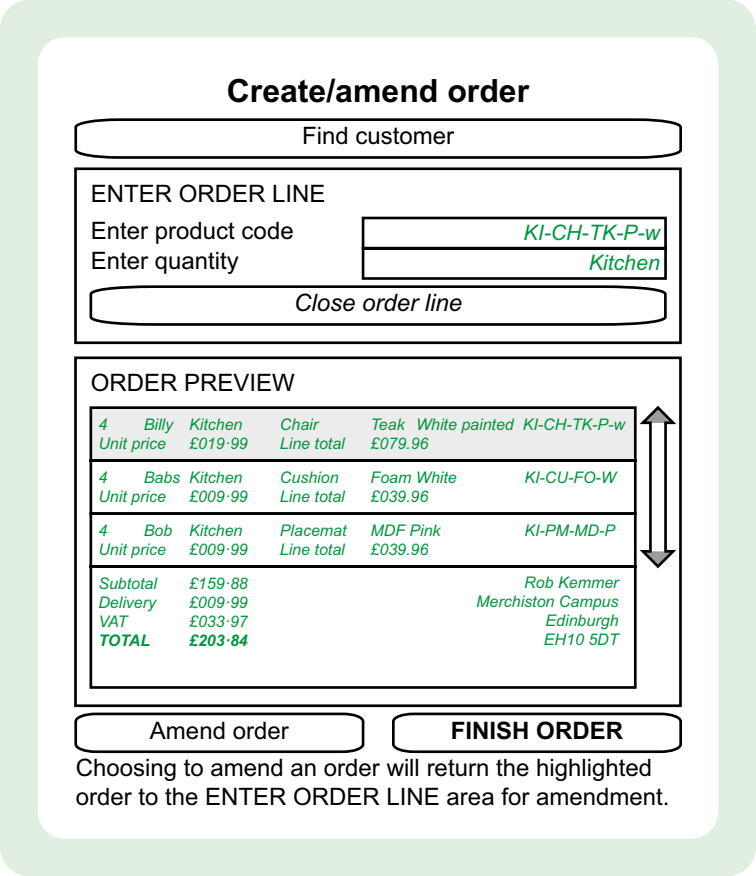
PHOTO *SUGGESTED*

Change to SUGGESTED

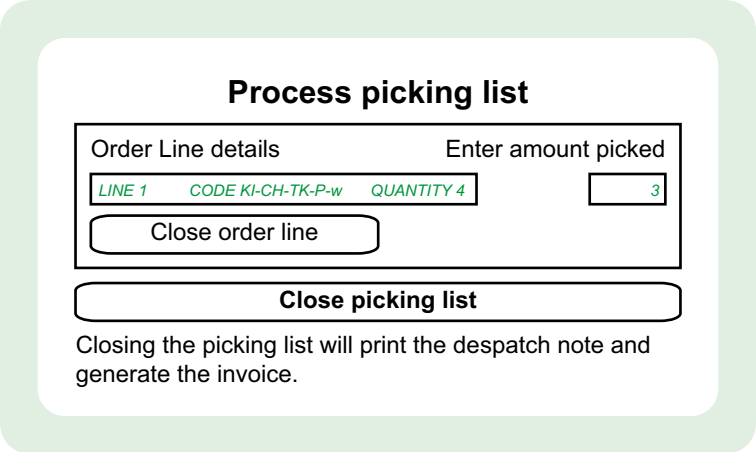
Choose file to upload

New photos are automatically 'suggested'.

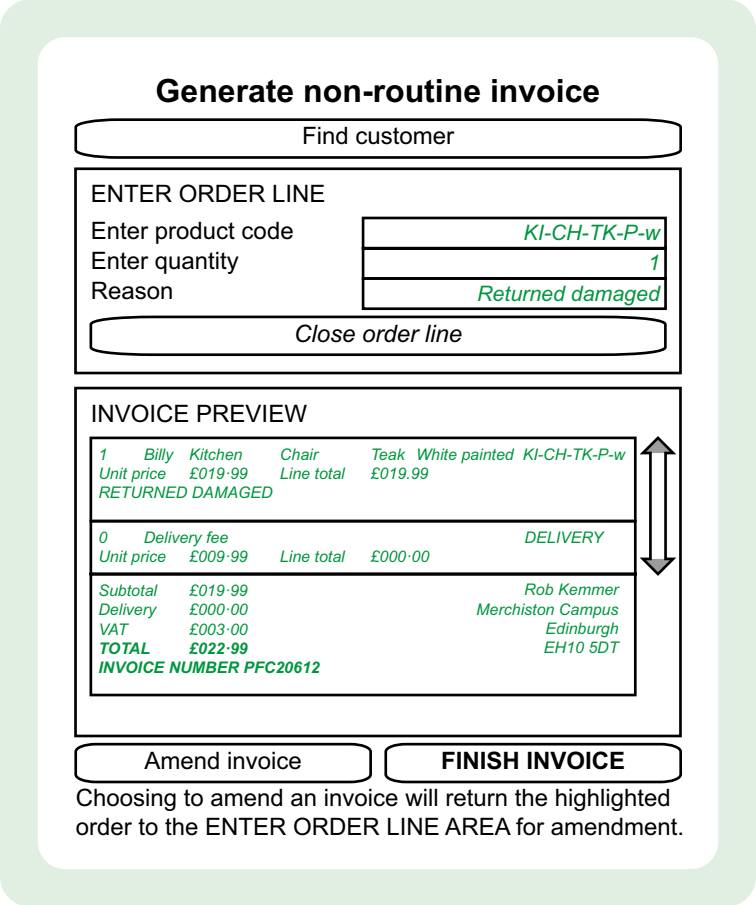
5.5 Enter order



5.6 Process picking list

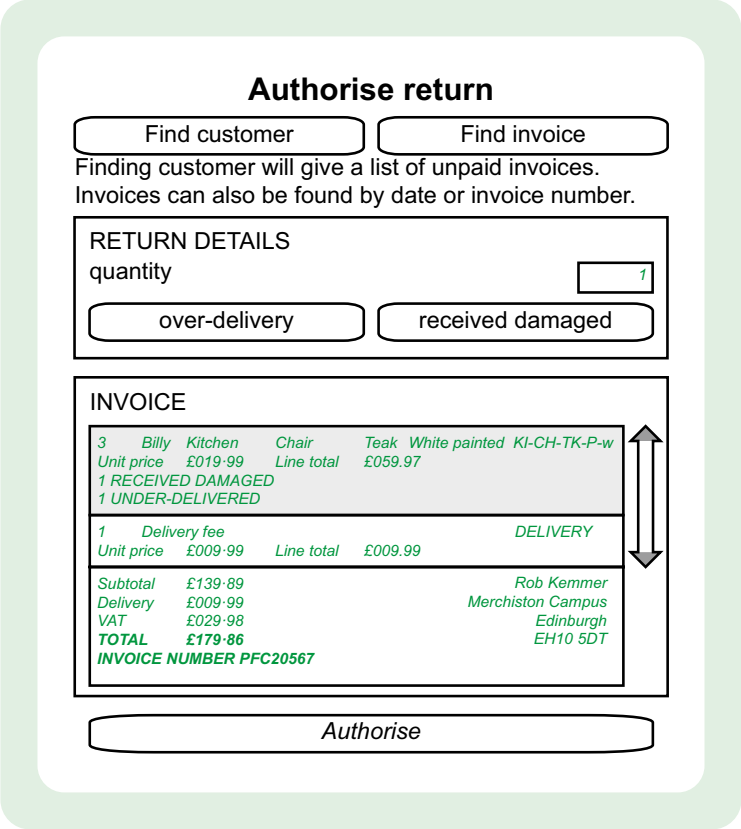


5.7 Generate non-routine invoice

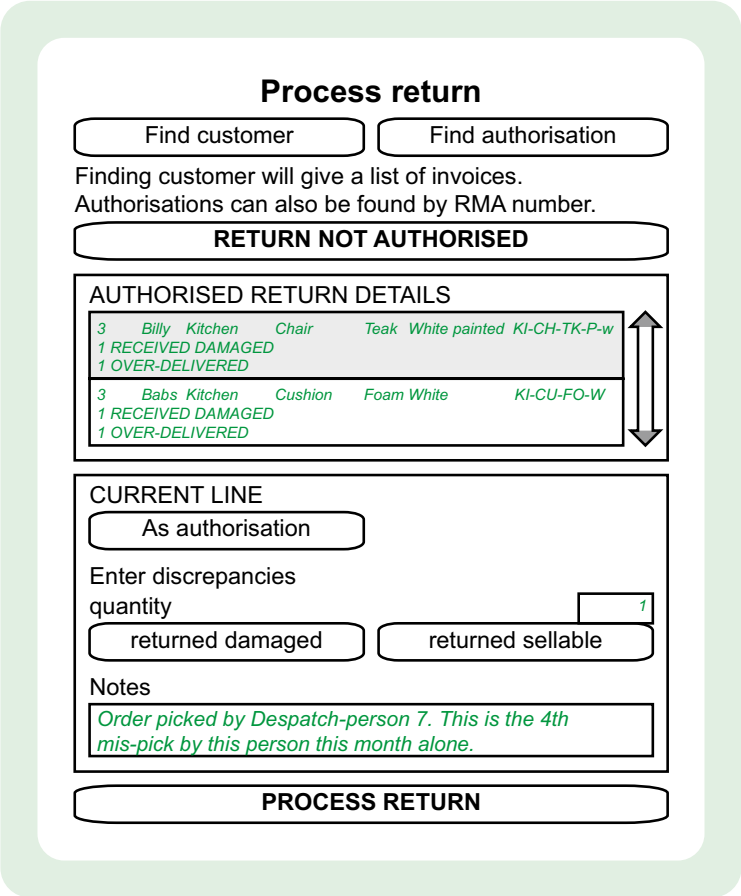


5.8 Enter payment

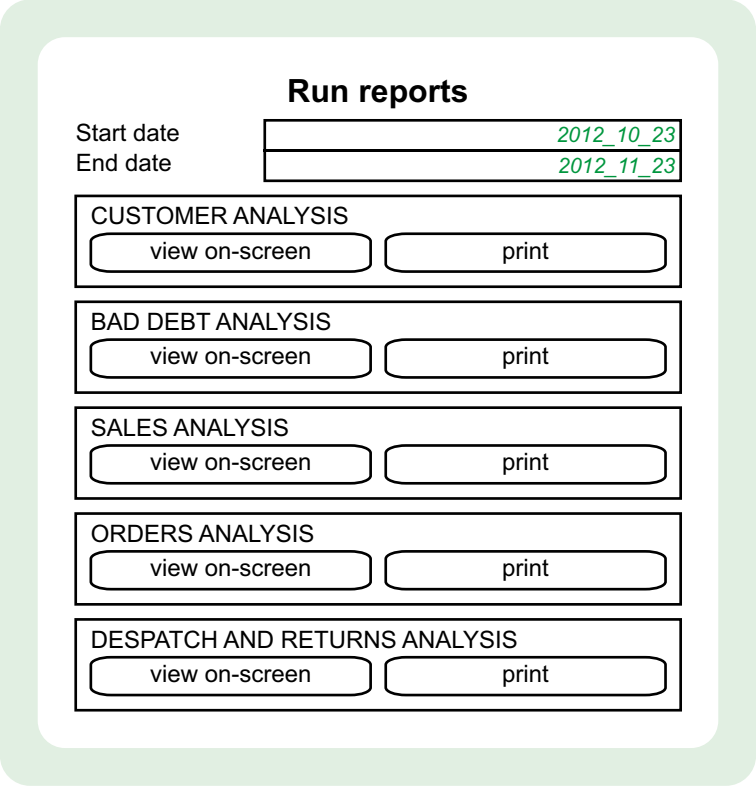
5.9 Authorise return



5.10 Process return



5.11 Run reports



6 Works Cited

Beck, K. et al. (2001). *Manifesto for Agile Software Development*. Retrieved November 23, 2012, from http://agilemanifesto.org.

Crashplan. (2012, November 09). *Frequently asked questions*. Retrieved November 21, 2012, from http://www.crashplan.com/business/support/doku.php/faq.

Ganeshan, D. (2011, June 06). *Waterfall versus Agile: a pros and cons analysis*. Retrieved November 23, 2012, from http://www.theserverside.com/tip/Waterfall-versus-Agile-methods-A-pros-and-cons-analysis.

Phatak, O. (2010, October, 15). What is agile model? Retrieved November 23, 2012, from http://www.buzzle.com/articles/what-is-agile-model.html.

Royce, W. W. (1970). *Managing the development of large software systems*. Retrieved November 23, 2012, from http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf.

Sonora Corporation. (2007, February 12). *GFS Backup Tape Rotation System*. Retrieved November 21, 2012, from http://www.sonoracomm.com/support/18-support/159-gfs.

Waterfall Model. (2012, October 7). *V Model*. Retrieved November 23, 2012, from http://www.waterfall-model.com/v-model-waterfall-model/.