

The ClearSpecs Challenge

David Gelperin,
ClearSpecs Enterprises

The Challenge

1. Specify the merchandise return capability (described below) using **your current techniques** or any others.
2. Have the ClearSpec Requirements (below) and your specs judged by a panel of stakeholders who need the requirements information (i.e. project estimators product architects, product testers, and technical communicators)

The Example

Consider the functional requirements for the merchandise return capability within a sales support system. There is a spectrum of specification precision from Title to Detail as follows:

Title	Merchandise Return Support Capability
Summary	Sales system must support merchandise returns. Missing receipts and manager overrides should be addressed.
Detail (natural language)	Sales system must support returns of unopened, unused, or defective merchandise for up to 14 days from date of purchase. With a valid sales receipt, returns must be accepted at any store. Without a valid sales receipt, returns must be accepted only at the store where purchased, and then only if the customer's sales record is located in the system and the customer provides a valid photo-id. Any store manager can override these rules and accept a return. [Etc.]
Detail (semi-formal language)	[Below, we show how an analyst might express these details using <i>ClearSpecs</i> techniques, starting with term definitions and action contracts, followed by precise use cases and then using test specs.]

When specifying this Merchandise Return (MR) capability, the project glossary would already be partially populated, because MR is unlikely to be the first capability described. These definitions would include stakeholders, non-stakeholder info sources, inter-actors, application tasks, essential classes, and system functions.

The analyst would add *return days limit* defined as the (arbitrary) constant 14 as a **function limits** type definition. Also, because there are six decision factors, the analyst would add *valid return* and *invalid return* as a **derived conditions** type definition as follows:

For the function – Merchandise Return,
valid return is any of:

After sale interval	Receipt status	Return location	Sales record status	Photo-id status	Manager override
≤ return days limit	valid				not given
≤ return days limit	invalid or missing	purchase location	found	valid	not given
					given

invalid return is any of:

After sale interval	Receipt status	Return location	Sales record status	Photo-id status	Manager override
> return days limit					not given
≤ return days limit	invalid or missing	not purchase location			not given
≤ return days limit	invalid or missing	purchase location	not found		not given
≤ return days limit	invalid or missing	purchase location	found	invalid or missing	not given

The analyst would add *after sale interval* as a **derived values** type definition as follows:

For the function -- Merchandise Return,
after sale interval = (current date – sale date)

Then the behavior of the Merchandise Return capability could be specified with **action contracts** as follows:

Recording customer info:

Constants		Post-conditions
Return status	Receipt status	Customer information
valid	invalid or missing	recorded

Recording return details:

Constants	Post-conditions
Return status	Return information
valid	details recorded

Forms of compensation:

Constants		Post-conditions
Return status	Compensation request	Compensation
valid	refund	payment refunded
valid	replacement	item replaced or payment refunded

Responses:

Constants	Post-conditions
Return status	Response
invalid	apologized for inconvenience

Suggestions:

Constants					Post-conditions
Return status	After sale interval	Return location	Sales record status	Photo-id status	Suggestion
invalid	\leq return days limit	not purchase location			try to find sales slip or return at purchase location
invalid	\leq return days limit	purchase location	not found		try to find sales slip
invalid	\leq return days limit	purchase location	found	invalid or missing	try to find sales slip or bring valid photo-id

Dispositions of returned item:

Constants		Post-conditions
Return status	Returned item type	Returned item "to be" disposition
valid	defective	returned to manufacturer
valid	unopened	reshelved
valid	unused	repackaged, put on sale, or returned to manufacturer

Then, the analyst might specify usage of this capability with a **precise use case** as follows:

Case Name: Return purchased items

Risk Factors: Frequency of occurrence: 0 to 3 per 100 transactions (average)
10 to 15 per 100 transactions (peak)

Impact of failure: *likely case* – **low**, if receipt, save copy for entry when function available
worst case – **low**, if no receipt, ask to return when function available

Case Conditions:

Constants: None

Preconditions: Customer has items to be returned

Interactions

Basic Course:

Customer	Sales System
1. requests merchandise return	2. requests sale info
3. provides sales a. receipt or b. date and location	4. accepts return, unless a. return is not valid <i>Preconditions</i> For sale, return is valid
	5. requests compensation preference
6. selects a. refund or b. replacement	7. provides compensation, records return details and thanks customer for their business <i>Post-conditions</i> For sale, items returned compensation provided return details recorded SUCCESS EXIT

Alternative Courses:

Optional Actions (OA)

OA1. After 3, manager overrides rules

Constants:

Customer has items to be returned

Preconditions:

Return is invalid

Customer or clerk requests a manager

Manager decides to override return rules

Post-conditions:

For sale, return is valid

Override is recorded

Manager	Sales System
1. approves override	2. records override

OA2. Before 4, system records customer information

Constants:

Customer has items to be returned

For sale, return is valid

Receipt is missing or not valid

Post-conditions:

Customer info is recorded

Customer	Sales System
	1. requests customer info
2. provides name, address, phone, and email	3. records customer info

Exception Handlers (EH):

EH 1 - 4a (return is not valid)

Constants:

For sale, return is not valid

Customer	Sales System
	1. Apologizes for inconvenience
	2. If (after sale interval > return days limit) FAILURE EXIT Endif
	3. Suggests trying to find the sales receipt
	4. If (not purchase location) also suggests returning to purchase location Else If (sales record found) also suggests returning with a valid photo-id Endif FAILURE EXIT

Finally, the analyst might specify **test specs** for the capability as follows:

Id: ATP 2.10 - Successful Merchandise Return

Objectives & Scope: To acceptance test successful merchandise return scenarios.

Special setup steps: None

Constant Conditions:

For sales system, status is active

For sale, return is valid

Pre-conditions:

Customer has items to be returned

Run &/or Check steps: Customer --

1. requests merchandise return

2. provides sales information

OA2.2. provides customer information, if asked

6. selects compensation

Post-conditions:

For sale, items returned

compensation provided

return details recorded

customer information recorded, if receipt not valid

override recorded, if manager decided to override

Special wrap-up steps: None

Values of scenario variables:

Test Case Id	Sales info	Mgmt override	Form of compensation	Purpose of test values
TC1	valid receipt	no	replacement	normal return
TC2	date & location	no	refund	valid return without receipt
TC3	date & location	yes	replacement	override invalid without receipt
TC4	valid receipt & > return days limit	yes	refund	override too late