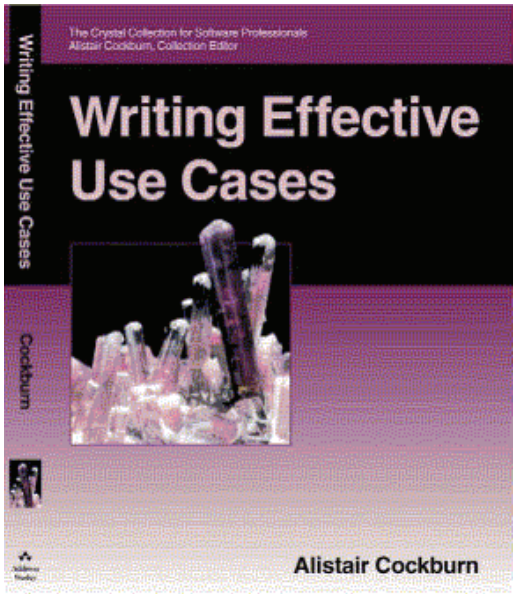
Writing Effective Use Cases

Dr. Alistair Cockburn
http://Alistair.Cockburn.us



Use case= text of scenarios of user succeeding or failing to achieve a goal using the system.

(goal of primary actor)

(level of goal [*summary* | *user* | *subfunction*])

(primary actor)

"Place an order" (User-level goal for 'Clerk')

(action steps: full sentences showing who takes the action! 3 - 9 steps.)

Main scenario:

1. Clerk identifies customer, item and quantity.
2. System accepts and queues the order.

(condition causing different actions)

(action steps to handle that condition)

Extensions:

1a. Low credit & Customer is 'Preferred':
System gives them credit anyway.

1b. Low credit & not 'Preferred' customer:
Clerk accepts only prepayment.

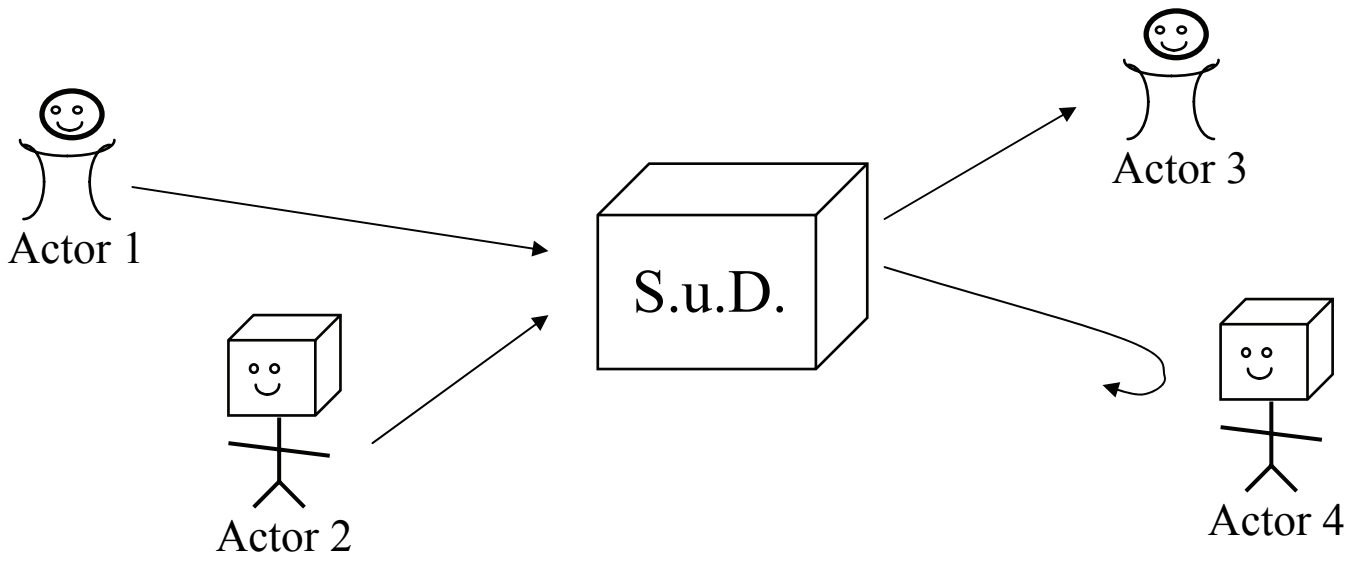
2a. Low on stock: Customer accepts rain-check:
Clerk reduces order to available stock level.

How to write a use case

How to do it:
(1). Capture the system boundary.

What ***computers***, ***subsystems*** and ***people*** ("Actors") will interact directly with our system?

Draw this as a picture:



How to do it:
(2). Identify the actors and their goals.

An "actor" is anything with behavior. Use the actors captured in step 1.

What does each actor want/need our system to do?

Result: a table listing of actors & use case names (user goals).

: Short complete list of usable system function.

: Not much detail, but very useful.

☺ Goals make a good structure on which to hang requirements & project details.

Project planning capitalizes on goal structure:

- : Useable Releases.
- : Priorities,
- : Schedule, staffing

Name	P. Actor	Pr.	Diff.	Release
Update customer	Customer	high	med	1
Scan products	Customer	high	high	1
Generate invoice	Finance	high	high	3
Funds transfer	Finance	med	high	4

(Note: spreadsheets are perfect for this!)

For each use case...
(3). Write the simple case: goal delivers.

The main success scenario, the happy day case.

: Easiest to read and understand.

: Everything else is a complication on this.

Capture each actor's intent and responsibility, from trigger to goal delivery.

: Say what information passes between them.

: Number each line.

Result: readable description of system's function.

"Place an order"

***Main scenario*:**

1. Clerk identifies customer, item and quantity.

2. System accepts and queues the order.

How to do it:
(4). Capture conditions needing other handling.

Usually, each step can fail.
WRite the failure condition ***after*** the main success scenario.

The extension scenarios show detecting unusual situations.

"What if their credit is too low?"
"What if they run over the extended credit limit?"
"What if we cannot deliver the quantity?"

Result: list of alternate scenarios to work out

Extensions:
1a. Low credit & Customer is 'Preferred':
1b. Low credit & not 'Preferred' customer:
2a. Low on stock: Customer accepts rain-check:

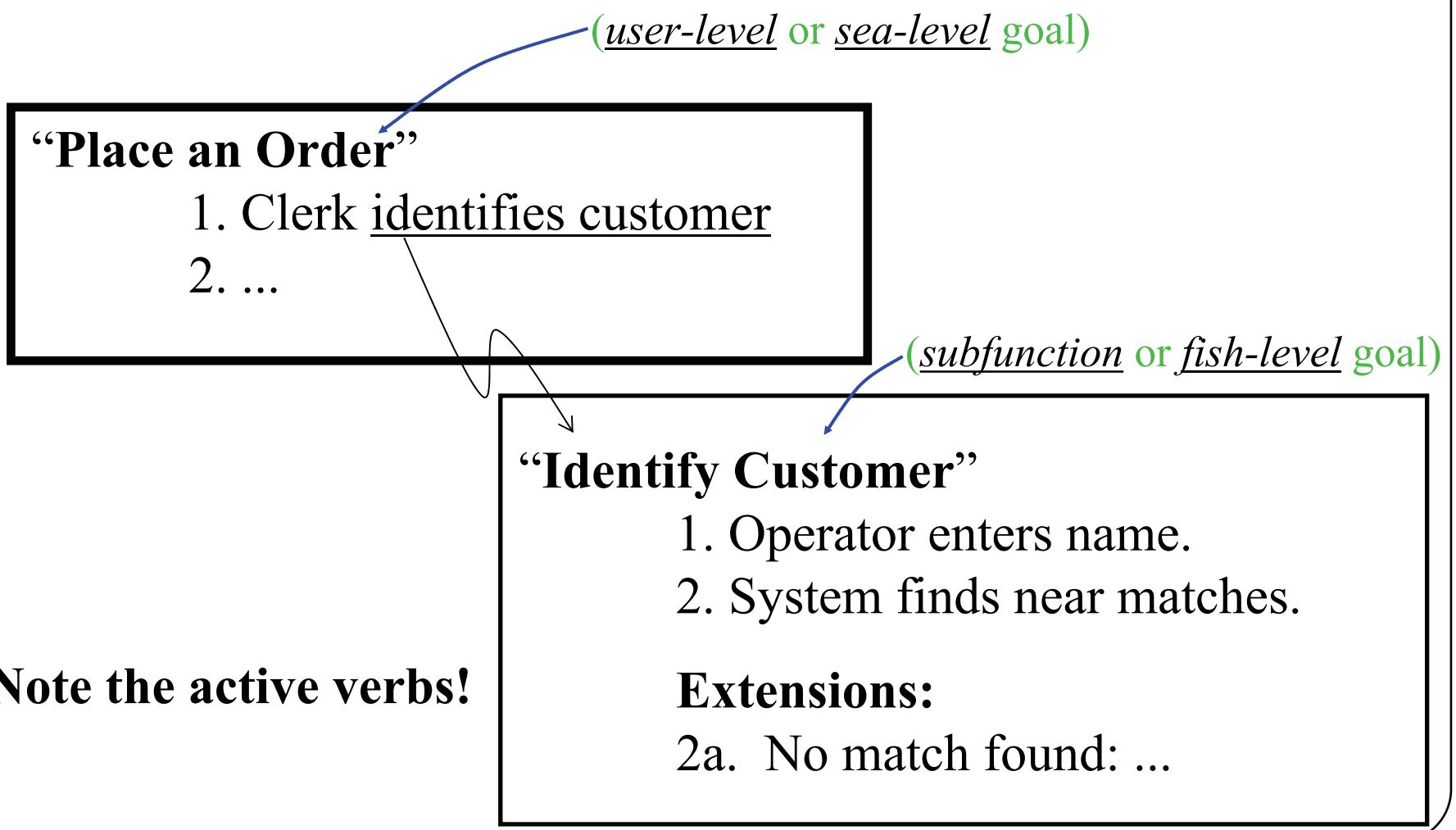
For each extension condition, (5). Follow the extension till it ends or rejoins.

Recoverable extensions rejoin main course.
Non-recoverable extensions fail directly.

Result: Complete use case

***Extensions*:**
2a. Low on stock: Customer accepts rain-check:
Clerk reduces order to available stock level.
2a. System crashes:
Use case fails

A scenario refers to lower-level goals (sub-use cases or common functions).



Note the active verbs!

The outer use case only cares if the inner one succeeds, avoiding proliferation.

UC 4: "Place an Order"

1. Clerk *identifies customer* <- (assumes success)
2. ...

Extensions:

1a. *Customer not found*: <-(does not care why it failed, only if it is recoverable)

The hard parts about use cases is not typing, but thinking and agreeing.

- ~ 3 days construction
  - : ~ 2 hours typing
  - : 2.75 days spent thinking & arguing about policies

1. Each step is correct.
   (Style: Each step written as "Actor kicks target.")
2. There are no missing system responsibilities between steps.
3. All outside systems this system should use is mentioned explicitly.
4. All stakeholders with valid interests have their interests met by the end.
5. Every extension condition needing attention is mentioned.

Good use cases do many good things for the project ...

1. Show stakeholders' interests in the system's behavior (the contract)
2. Show requirements in a context of use
3. Show record of requirements decisions made
4. Communicate in a language that crosses specialties
5. Allow check for completeness
6. Give advance notice of items needing research

COMMON MISTAKES: "**Register for Courses**"

(*Patterns for Effective Use Cases*, Steve Adolph & Paul Bramble, UC 1.1)

1. Display a blank schedule.
2. Display a list of all classes in the following way: The left window lists all the courses in the system in alphabetical order. The lower window displays the times the highlighted course is available. The third window shows all the courses currently in the schedule.
3. Do
4. Student clicks on a course.
5. Update the lower window to show the times the course is available.
6. Student clicks on a course time and then on the "Add Course" button.
7. Check if the Student has the necessary prerequisites and that the course offering is open.
8. If the course is open and the Student has the necessary prerequisites, add the Student to the course. Display the updated schedule showing the new course. If no, put up a message, "You are missing the prerequisites. Choose another course."
9. Mark the course offering as "enrolled" in the schedule.
10. End do when the Student clicks on "Save Schedule."
11. Save the schedule and return to the main selection screen.

Corrected "**Register for Courses**"

(*Patterns for Effective Use Cases*, Steve Adolph & Paul Bramble, UC 1.3)

System: Course Enrollment System Goal level: User Goal

1. Student requests to construct a schedule.
2. The system prepares a blank schedule form.
3. The system gets available courses from the Course Catalog System.
4. Student selects up to 4 primary and 2 alternate course offerings.
5. For each course, the system verifies that the Student has the necessary prerequisites, adds the Student to the course, marking Student as "enrolled" for that course in the schedule.
6. When the Student indicates the schedule is complete, the system saves it.

Extensions:

1a. *Student already has a schedule*: System brings up the current version of the Student's schedule for editing instead of creating a new one.

1b. *Current semester is closed and next semester is not yet open*: System lets Student look at existing schedules, but not create new ones.

3a. *Course Catalog System does not respond*: The system notifies the Student and the use case ends.

5a. *Course full or Student has not fulfilled all prerequisites*: System disables selection of that course and notifies the Student.

Bonus Pack

Use cases for agile development and short iterations

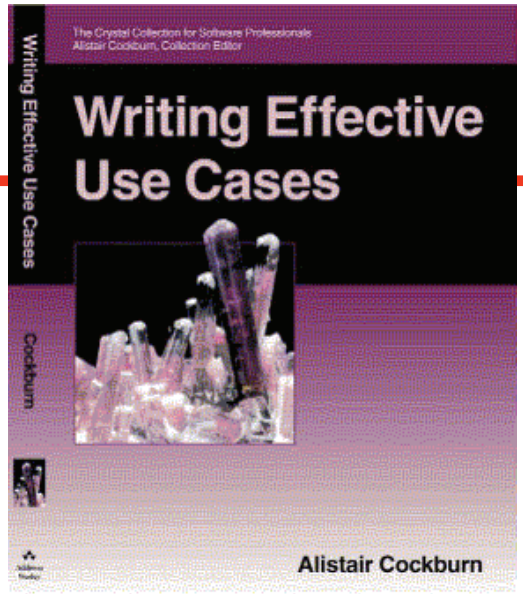
UC 7: "Register for Courses" ***Patterns for Effective Use Cases***, Adolph-Bramble)

System: Course Enrollment System Goal level: User Goal

[1] 1. Student requests to construct a schedule.
[1] 2. The system prepares a blank schedule form.
[2] 3. The system gets available courses from the Course Catalog System.
[3] 4. Student selects up to 4 primary and 2 alternate course offerings.
[4,5] 5. For each course, the system verifies that the Student has the necessary prerequisites, adds the Student to the course, marking Student as "enrolled" for that course in the schedule.
[1] 6. The Student indicates the schedule is complete, the system saves it.

("new function")
("more function")

Extensions:
[6] 1a. *Student already has a schedule*: System brings up the current version of the Student's schedule for editing instead of creating a new one.
[7] 1b. *Current semester is closed and next semester is not yet open*: System lets Student look at existing schedules, but not create new ones.
[8] 3a. *Course Catalog System does not respond*: The system notifies the Student and the use case ends.
[5] 5a. *Course full or Student has not fulfilled all prerequisites*: System disables selection of that course and notifies the Student.



Use Cases

Alistair Cockburn
Humans and Technology

acockburn@aol.com
http://Alistair.Cockburn.us