Avancier Methods (AM)
TECHNIQUES

Communication models

It is illegal to copy, share or show this document
(or other document published at http://avancier.co.uk)
without the written permission of the copyright holder

C**Communication modelling techniques**

A► *What follows are only a few notes on modelling notations and issues - not a treatise on the topic*

VAvancier

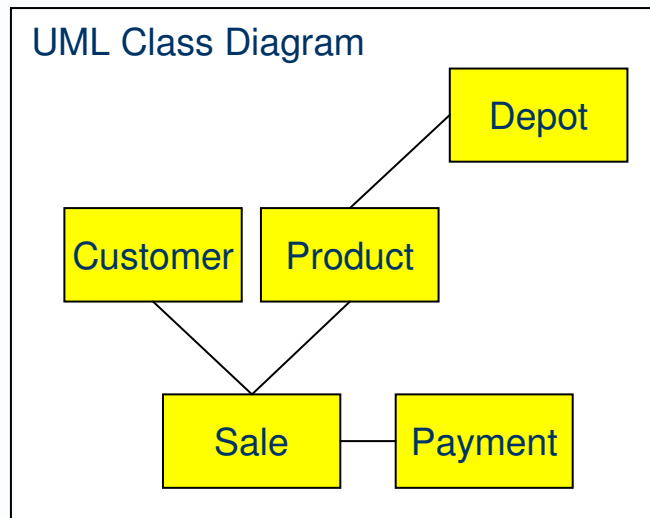# Using UML to model software structure and behaviour

One
Class diagram
Structural model showing
Module types (classes)
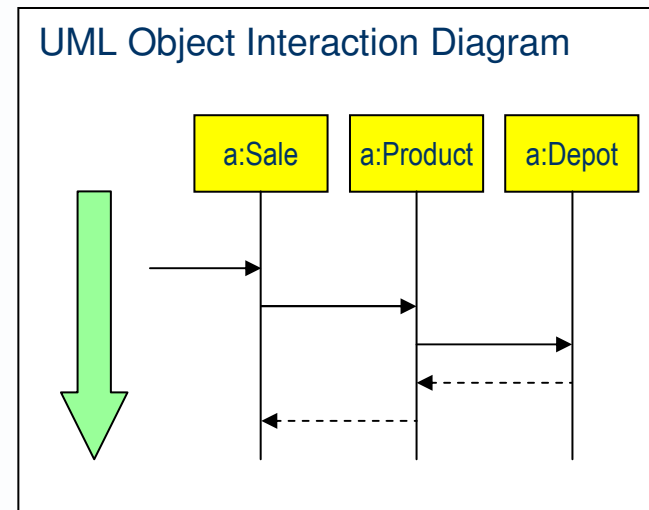Relationships between modules

UML Class Diagram

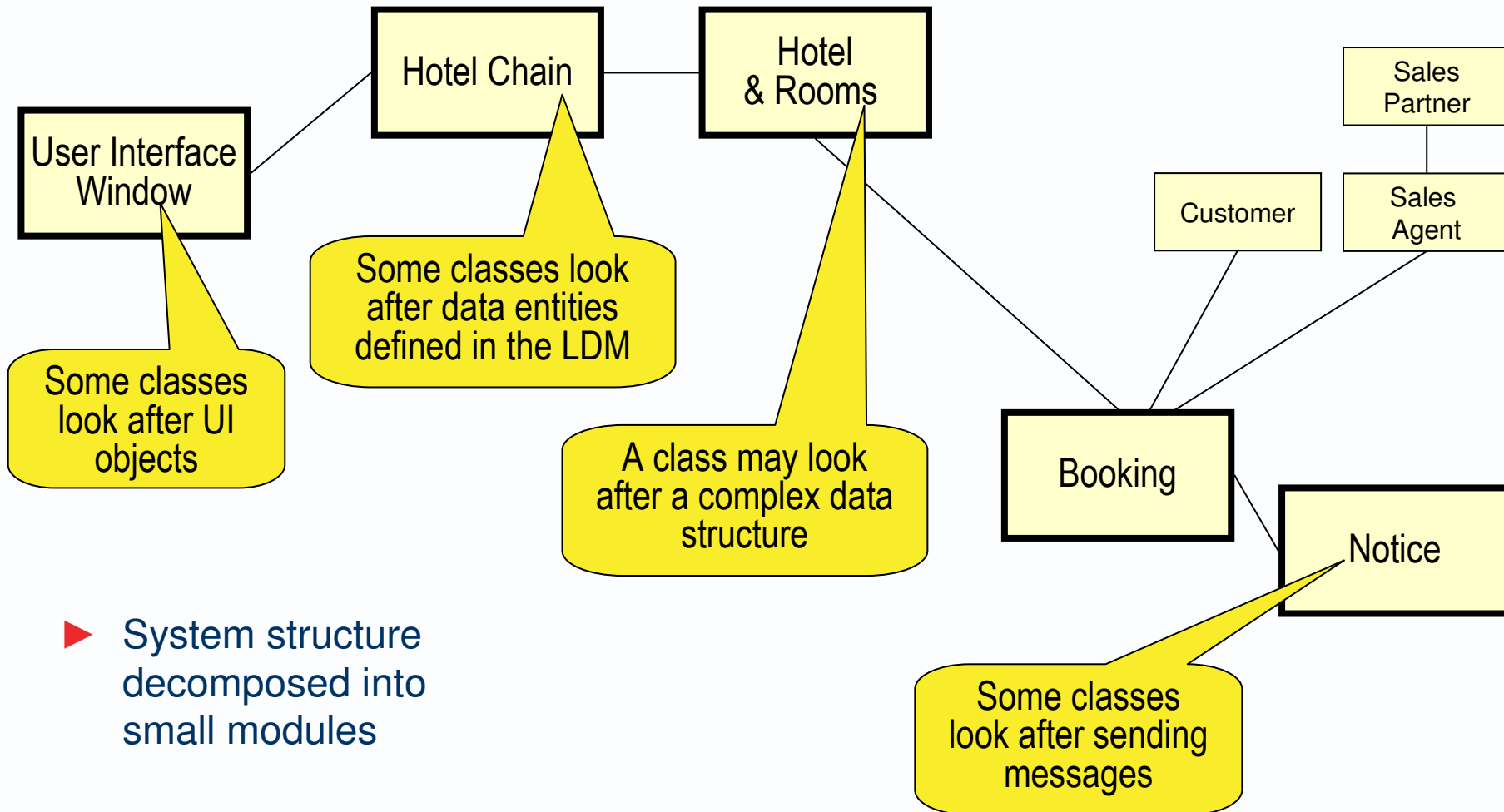Several
Interaction (sequence) diagrams
Behavioural models showing
How modules (objects) cooperate to
    perform a process

UML Object Interaction Diagram

A simple class diagram

Hotel Chain

Hotel & Rooms

User Interface Window

Sales Partner

Customer

Sales Agent

Booking

Notice

Some classes look after UI objects

Some classes look after data entities defined in the LDM

A class may look after a complex data structure

Some classes look after sending messages
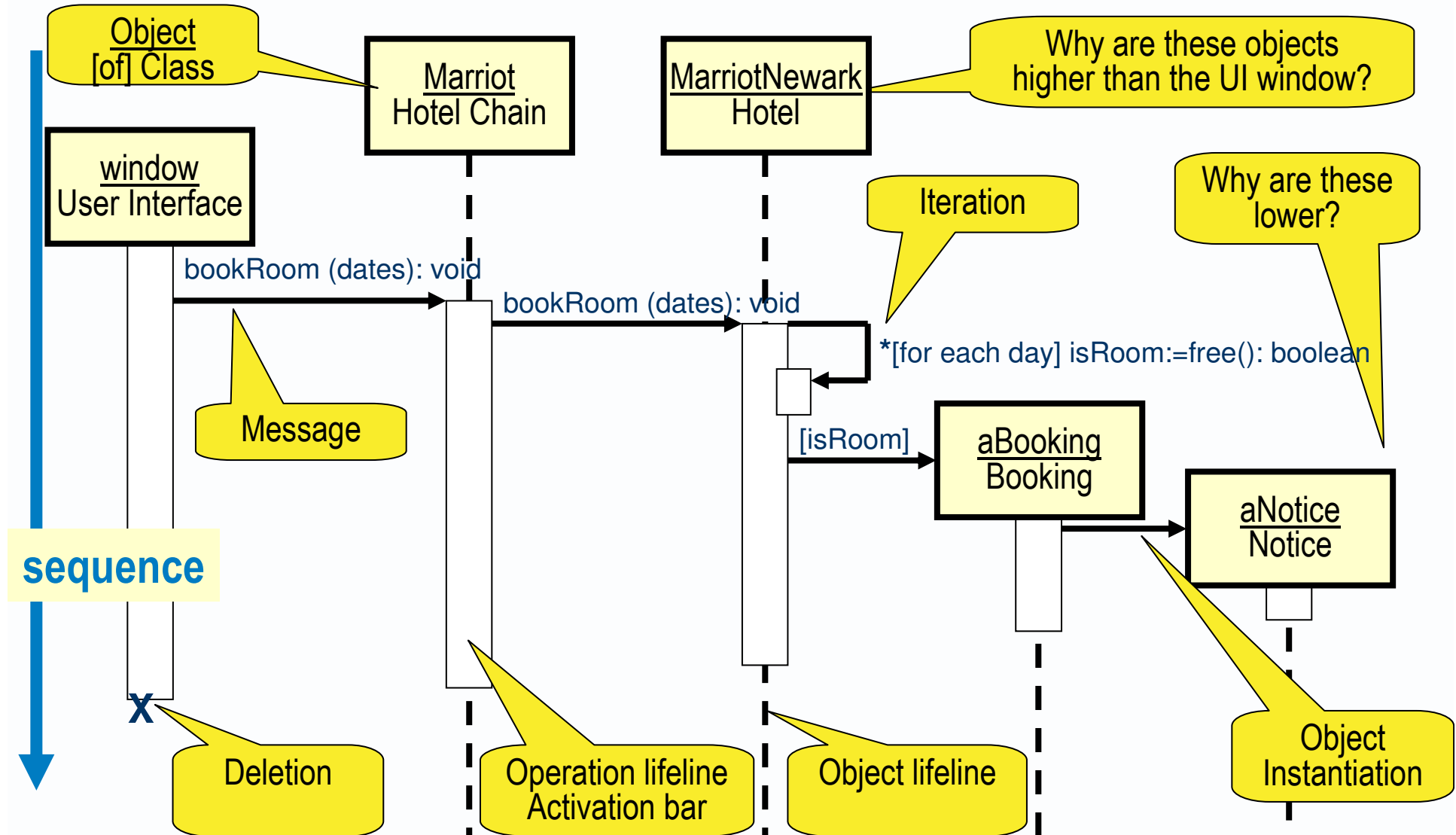
▶ System structure decomposed into small modules

How objects cooperate to perform a higher level process

Object [of] Class

Marriot Hotel Chain

MarriotNewark Hotel

Why are these objects higher than the UI window?

window User Interface

Iteration

Why are these lower?

bookRoom (dates): void

bookRoom (dates): void

*[for each day] isRoom:=free(): boolean

Message

[isRoom]

aBooking Booking

aNotice Notice

sequence

X

Deletion

Operation lifeline Activation bar

Object lifeline

Object Instantiation

Example drawn from somewhere on the internet

1. Request-reply means the client not only *waits* but also *freezes*

Operation (arg): reply

■ **Synchronous** ⟶ **Request-Reply**

■ **Asynchronous** **Operation (message)** ⟶ **Fire and Forget**

■ Strictly speaking, all human interaction is fire and forget

■ Even if, in practice, you mostly wait for a reply

Architects use of sequence diagrams

1. More *sketch* than detailed software design

2. More *coarse-grained components* than small OOP classes

3. More *asynchronous* inter-component communication

■ **Synchronous** — — — — — — → **Request-Reply**

Operation (arg): reply

■ **Asynchronous** ————→ **Fire and Forget**

Operation (message)

■ Which means *you cannot show operation life times!*
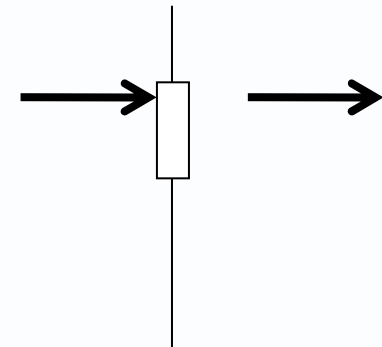
Process flow in a UML sequence diagram

The same process flow in an Event Impact Diagram
(Effect Correspondence Diagram in SSADM)

▶ **Nodes:** all the entities affected by an event
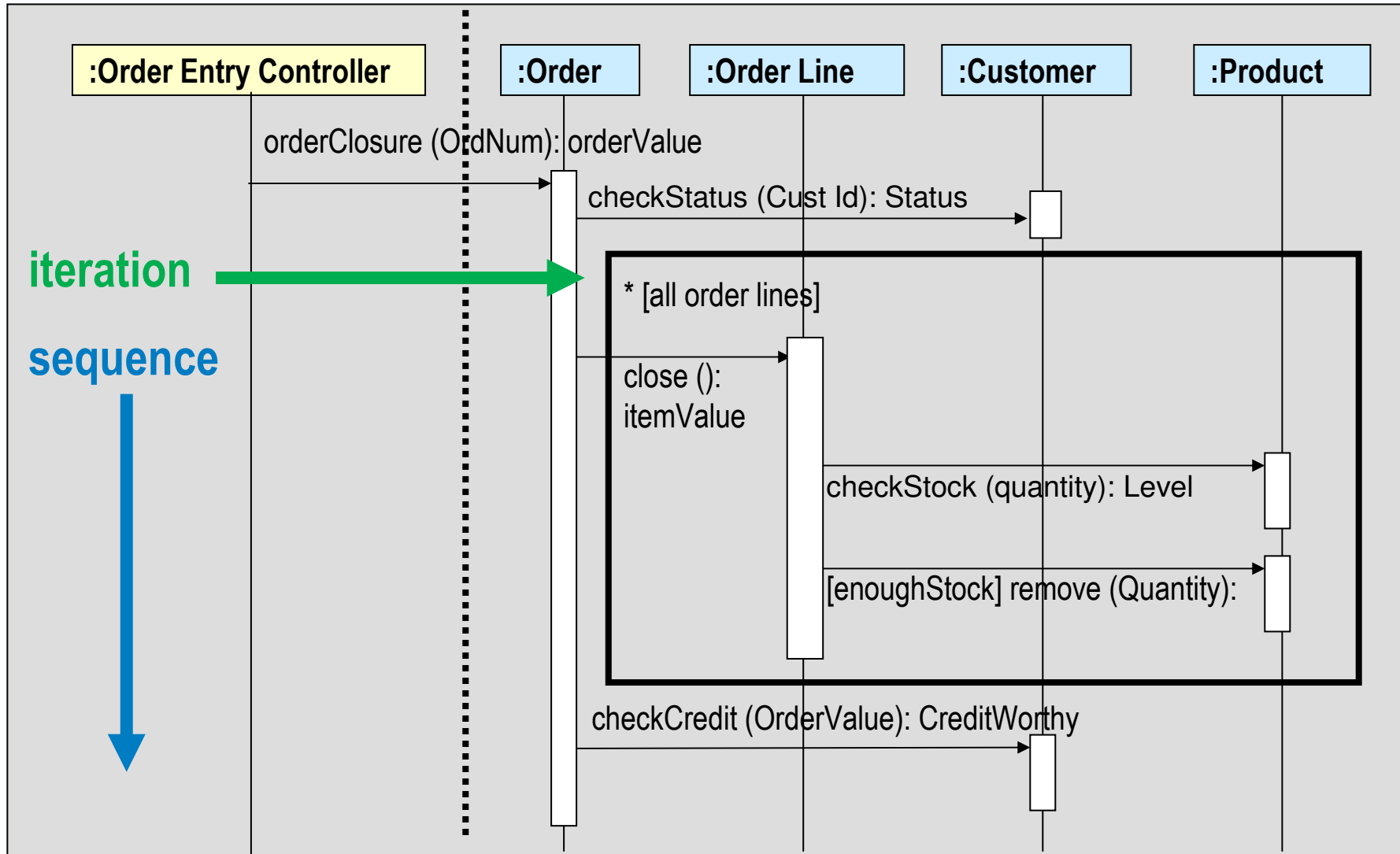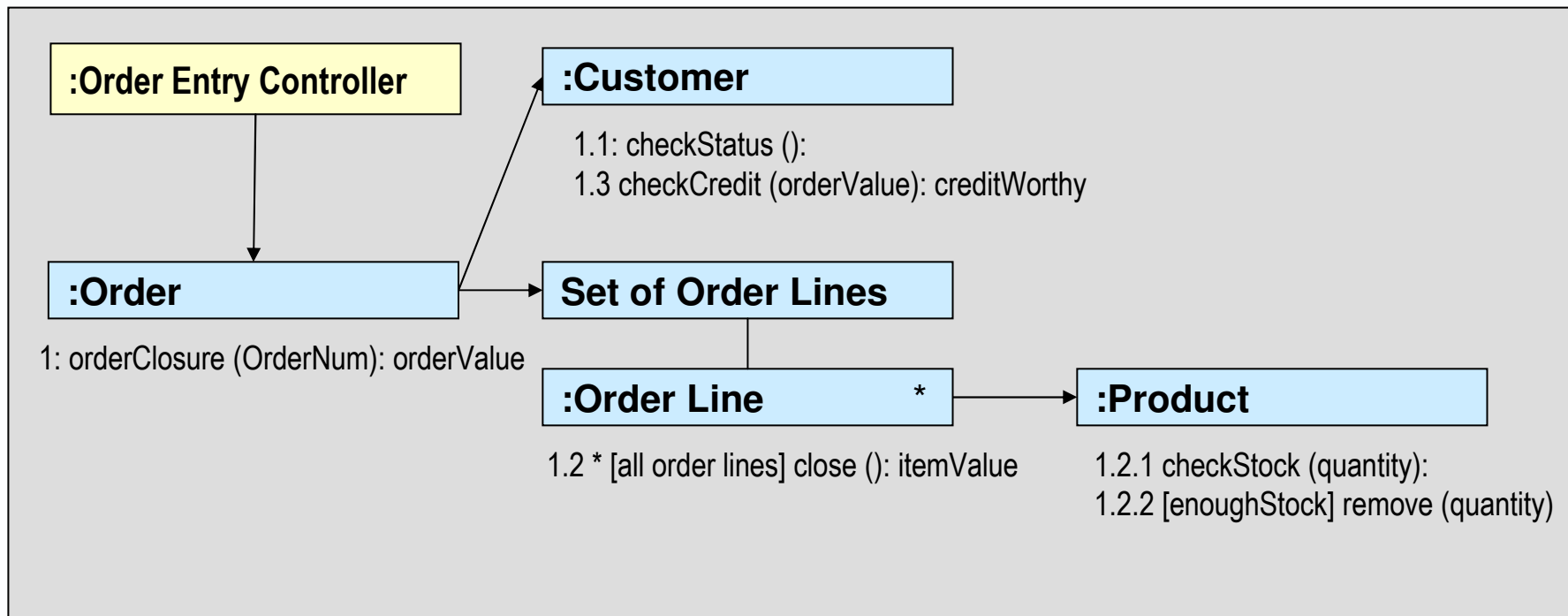
▶ **Lines:** 1-to-1 navigation from one entity to another entity or set of entities

An Event Impact Diagram
(Effect Correspondence Diagram in SSADM)

▶ This notation shows the effect one event has one or more entities

▶ The arrows are one-to-one correspondences

▶ IF the entities are coded as separate objects, THEN it is likely the arrows turn into messages

Button Push

What JSD would call a 'context filter'

A gatekeper object at the entry point of the event, not updated by this event

Oven-Door

Oven-Door ` SV = 'open'

Oven-Door (no effect)

Oven-Door (effect)

Oven-TBP

Oven-TBP ` SV = ' cooking'

Oven-TBP (extend cooking)

Oven-TBP (start cooking)

Oven-Light (start cooking)

Operations List

1 Set Timer for 1 minute
2 Add 1 minute to Timer
3 Energise Power tube
4 Invoke §Light On, and Fail If §Light On Fails
5 Set Oven-TBP ` SV = ' cooking'
6 Fail Unless Oven-TBP ` SV = ' idle'

2   6 1 3 5   4

From Event Impact Diagram to Interaction Diagram

Button Push

What JSD would call a 'context filter'

A gatekeper object at the entry point of the event, not updated by this event

Oven-Door

Oven-Door ` SV = 'open'

Oven-Door (no effect)

Oven-Door (effect)

Oven-TBP

Oven-TBP ` SV = ' cooking'

Oven-TBP (extend cooking)

Oven-TBP (start cooking)

Oven-Light (start cooking)

Operations List

1. Set Timer for 1 minute
2. Add 1 minute to Timer
3. Energise Power tube
4. Invoke §Light On, and Fail If §Light On Fails
5. Set Oven-TBP ` SV = ' cooking'
6. Fail Unless Oven-TBP ` SV = ' idle'

(2) (6) (1) (3) (5) (4)

UI Layer	Button Push	Door	Oven TBP	Light

Button Push

[Door closed]   [Cooking]

2

6 1 3 5

4

1. Set Timer = 1 Minute
2. Add I Minute to Timer
3. Energise Power Tube
4. Invoke $Light on
5. Set Oven TBP SV = Cooking
6. Fail unless Oven TBP SV = Idle