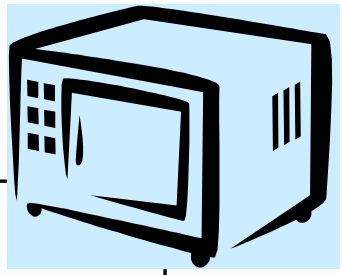
# Avancier Methods (AM)
## TECHNIQUES

## Client-Server Systems

# Single-user microwave simulator

Avancier

- The graphic below simulates a compact and inexpensive microwave oven.



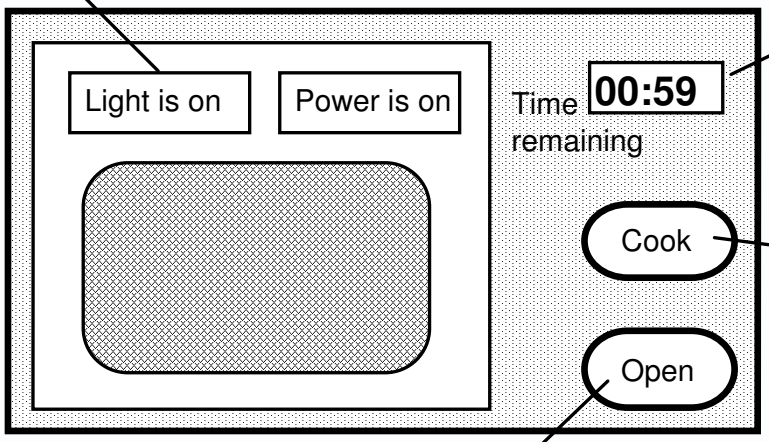
Aims

Goal

Objective

Requirement

The light inside the oven must be turned on while the oven is cooking (so you can see how the food is cooking) and while the door is open (so you can see to handle food and clean the oven).




When the oven timer reaches zero, it turns off the power tube, turns off the light and sounds the beeper to tell you the food is ready.

Push the cook button while the door is open - this has no effect.

Push the cook button while the door is closed - the oven starts to cook for a minute: that is, the power and the light come on, and the timer starts counting down from sixty seconds.

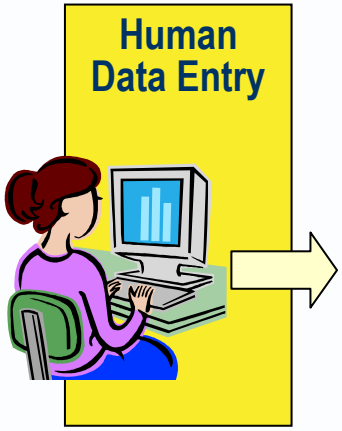
Push the cook button before the timer runs down - an extra minute of cooking time is added to the timer.

Open the door while the power is off - the light comes on. Close the door again - the light goes off.

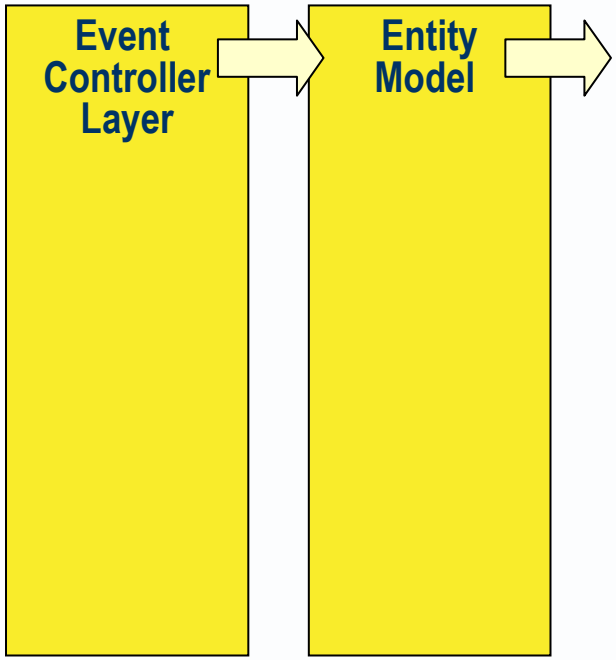
Open the door while the power is on - the power goes off and the timer returns to zero, but the light stays on. Close the door again - the light goes off, but cooking does not restart unless you push the cook button again.

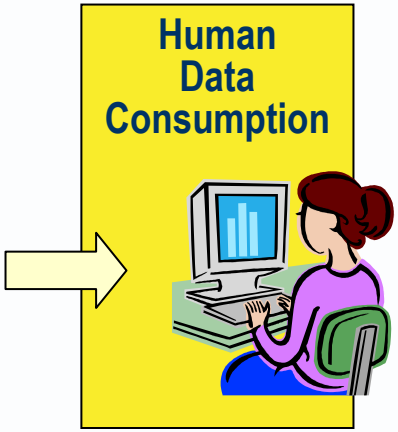
# The single-user simulator

Avancier




- The single-user microwave simulator (in the presentation “Entity event modelling”) has these main elements.



# What makes the single-user system simple?


- There is little data to be maintained and no persistent database
- There is no class hierarchy or inheritance structure
- The end-user does nothing but press control buttons on the screen.
- The input and output layer is readily designed and coded via prototyping a user interface.
- The event controllers are trivial.
- The input and output messages are trivial.
- All the code runs on one computer.
- The system is very small; the very few entities can be coded in a single program by one programmer.

- This allows us to ignore complexities that arise in real system design.

# How to build a multi-user client-server system?


- How to turn the model and the code written in a multi-user system in which
  - many users play with a microwave simulator on their local PC or lap top?
  - all the entities and events are recorded in a persistent database?

- We have to address
  - Data storage and retrieval
  - Model/Database structure clashes
  - Human data entry and data usage
  - Event controller classes
  - Input and output message transformations
  - Distribution of processing
  - Size and complexity

# Data storage and retrieval


Single-user microwaver

- There is little data to be maintained while the oven is on, and no persistent database

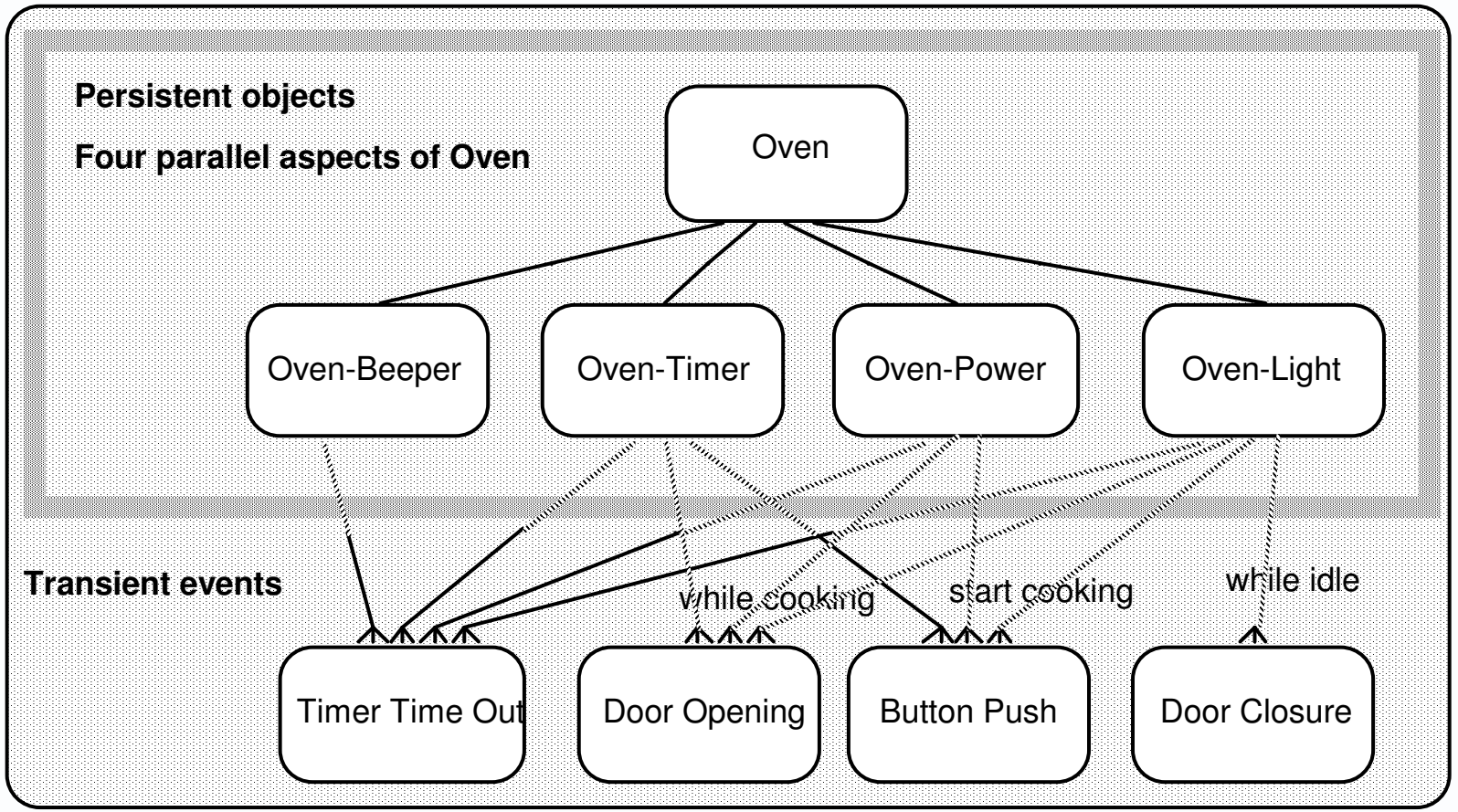
Multi-user client-server system

- Database design can take up much of the effort
- An entity relationship diagram is a major design document

- (Even in an embedded process control system, you can use an entity relationship diagram to show which events affect which objects.
  - If one persistent object is affected by many transient events, then you can draw one-to-many association between them.
  - If the event does not always affect the object, then you should make the association optional from the detail's point of view, only established if a condition applies.)

# Entity relationship diagram (inc events as entities)

- If you draw all one-to-many associations from the top-to-bottom, the transient events naturally appear at the bottom of the diagram (the “transient at the bottom” pattern).



# Model/Database structure clashes


Single-user microwaver

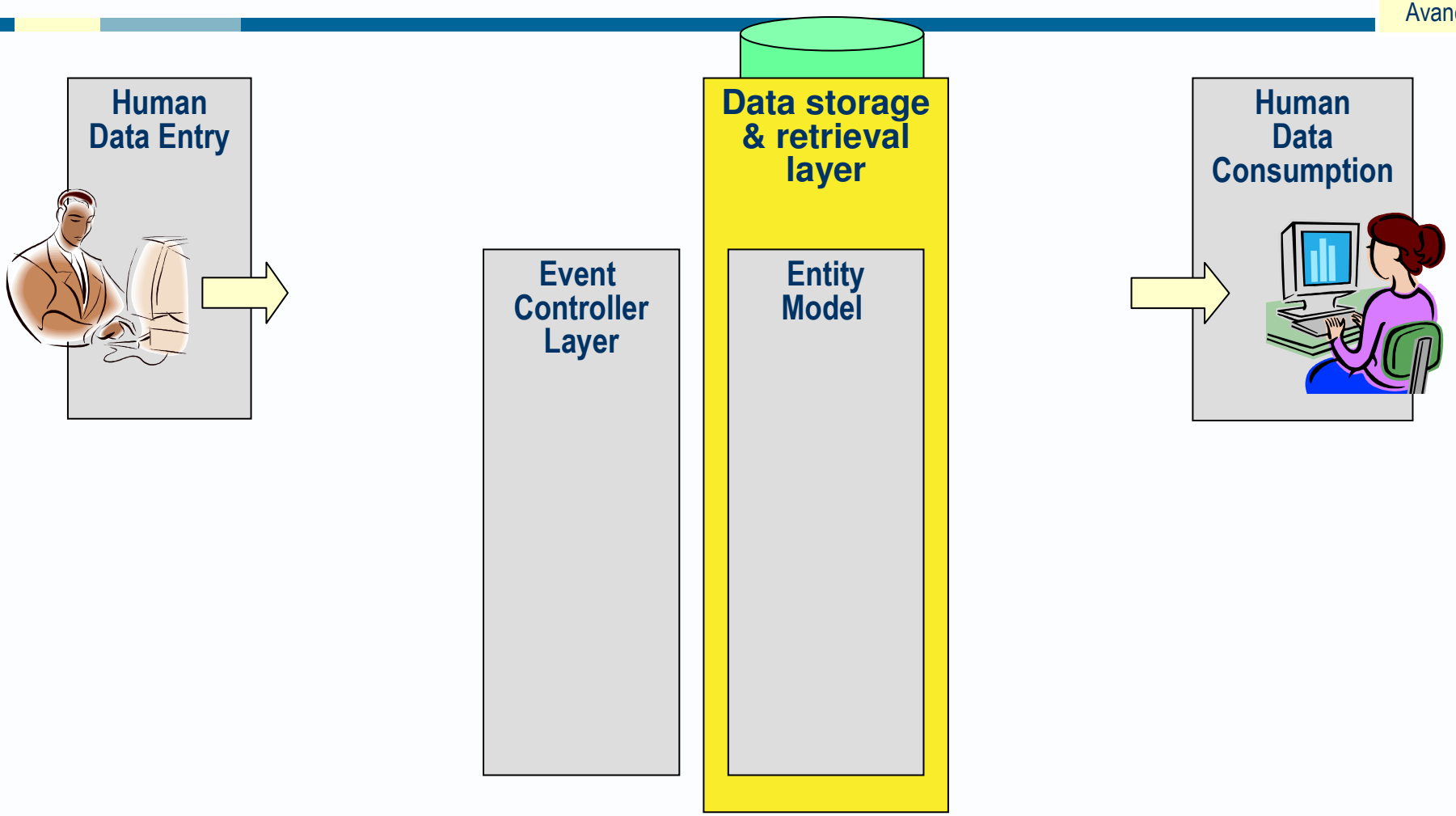
- There is no class hierarchy or inheritance structure
- A class hierarchy in the entity class model would reveal a special kind of structure clash that occurs between
  - models of object-oriented programs
  - models of persistent business data entities

Multi-user client-server system

- Often made very complex by the introduction of an “Model/Database mapping layer” between the OO program classes and the database tables.
- In some cases, it is said, this layer can consume 40 percent of the coding effort.

# A layer to handle database storage and retrieval

Avancier



# Human data entry and data usage

Avancier

Single-user microwaver

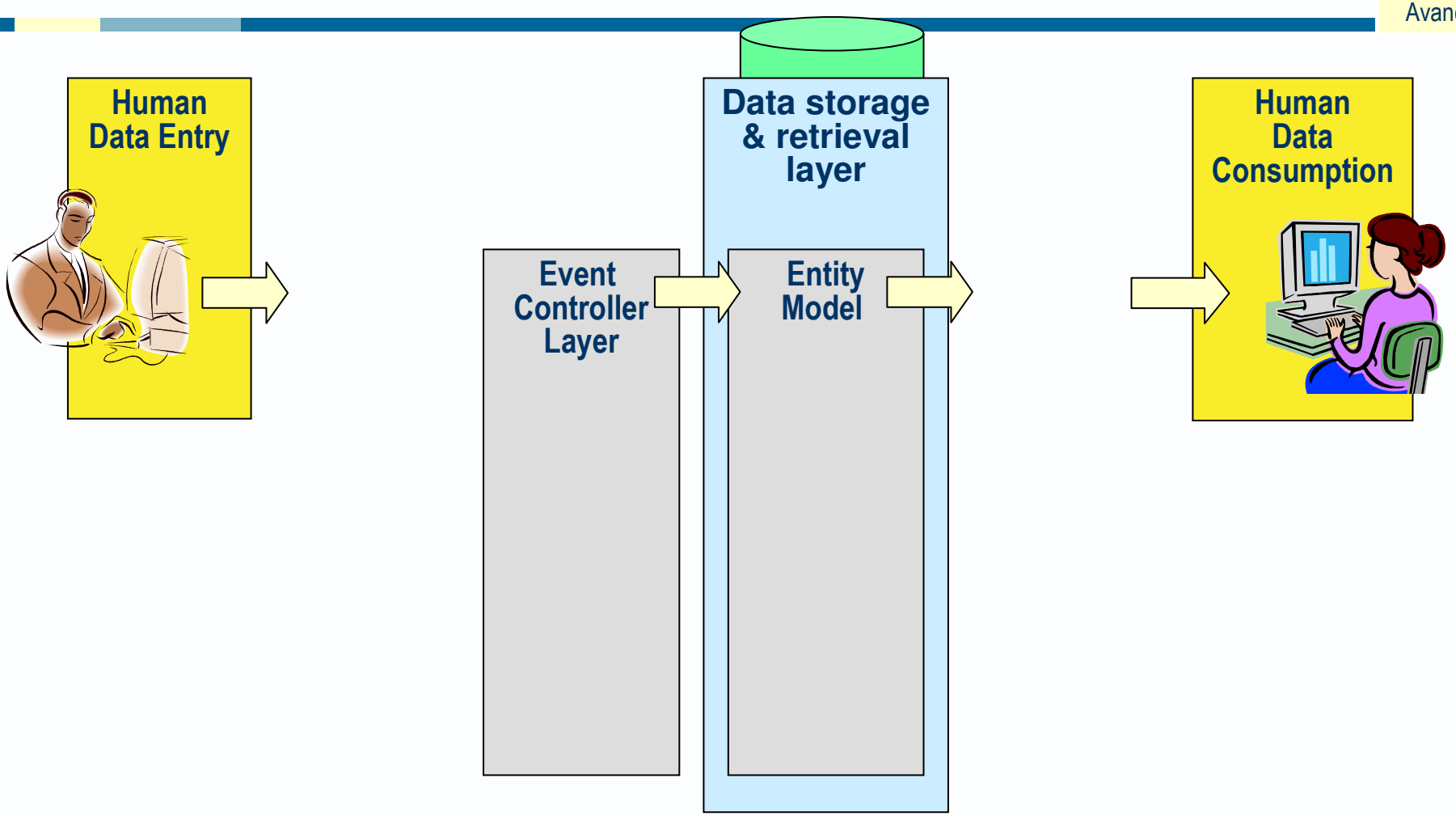
- The end-user does nothing but press control buttons on the screen.
- The input and output layer is readily designed and coded via prototyping a user interface.

- we drew a mock microwave Oven out of simple shapes, using Visual Basic to show the
  - Timer value in minutes and seconds,
  - Door (Open or Closed),
  - Light (On or Off), and
  - Power (Cooking or Idle).

Multi-user client-server system

- Human users have to make a considerable data entry effort
- The user interface can take up much of the design effort.

# Data entry and/or data usage layer(s)



# More complex event controller classes


Single-user microwaver

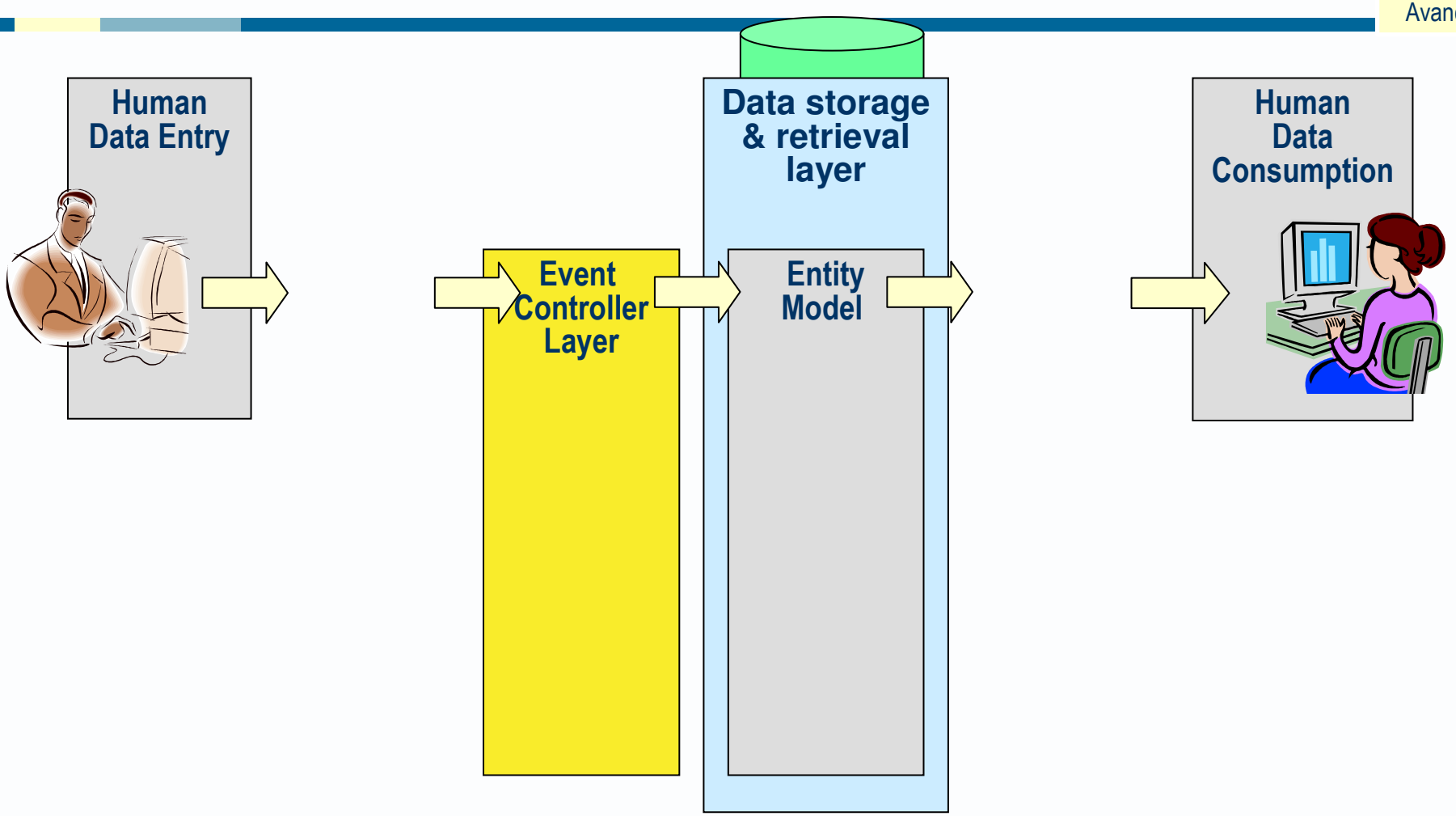
- The event controllers are trivial.
- The user interface input layer collects parameters, invokes an event class which does nothing but invoke the first object shown in its interaction diagram.

Multi-user client-server system

- Needs event controller classes to
- start transactions
- invoke one or more persistent objects (at least those that are identified by the event parameters).
- assemble the data structure that makes up the response to the event
- return or send this data structure for output
- commit/roll back transactions

# More complex event controller layer

Avancier



# Input and output message transformations

Avancier

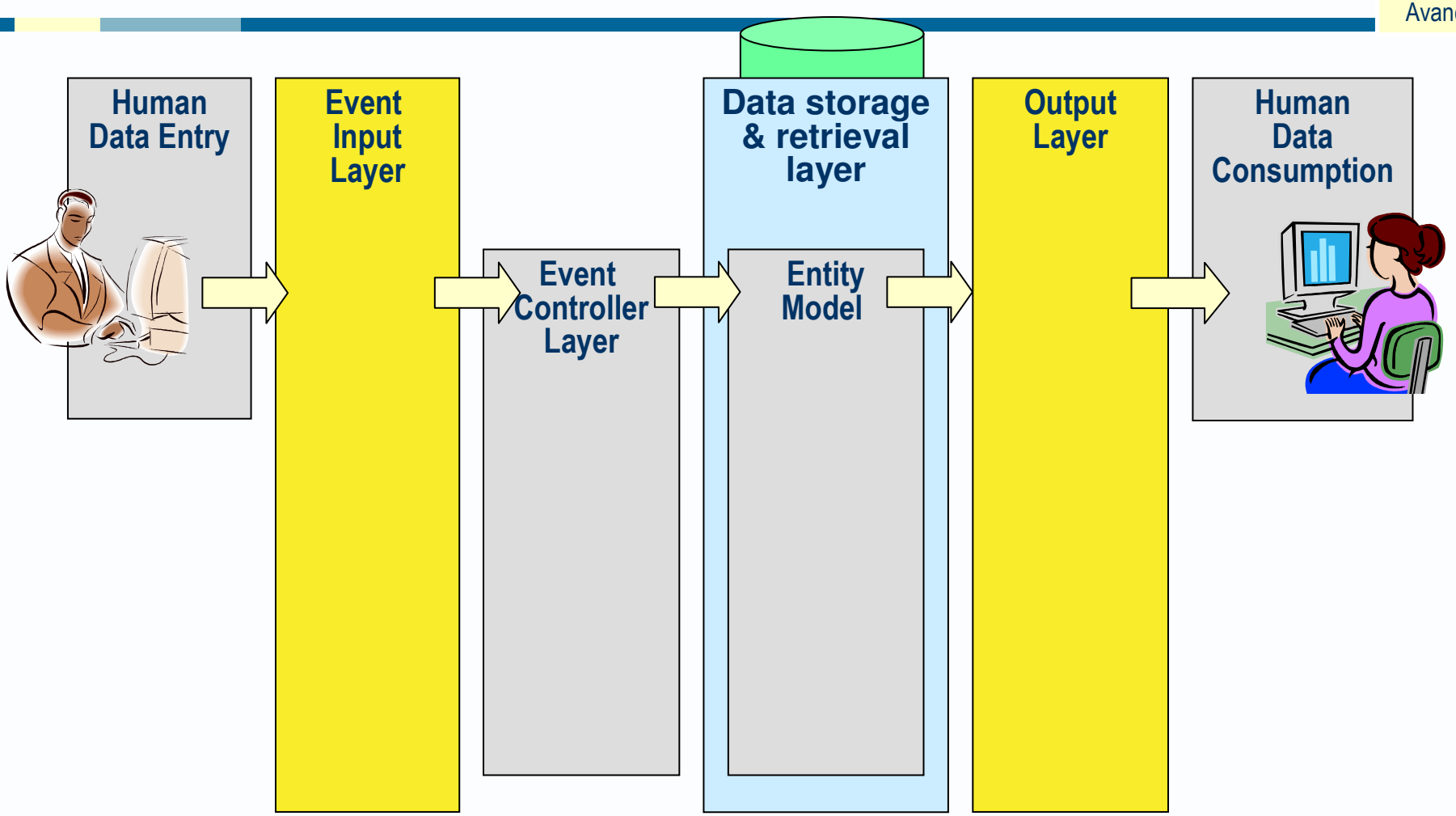
Single-user microwaver

- The input and output messages are trivial.

Multi-user client-server system

- Events can carry long and complex messages.
- Input and output data must be transformed from one form to another, data types and data structures must be changed

# Layers to transform input and output data



# Distribution of processing


Single-user microwaver

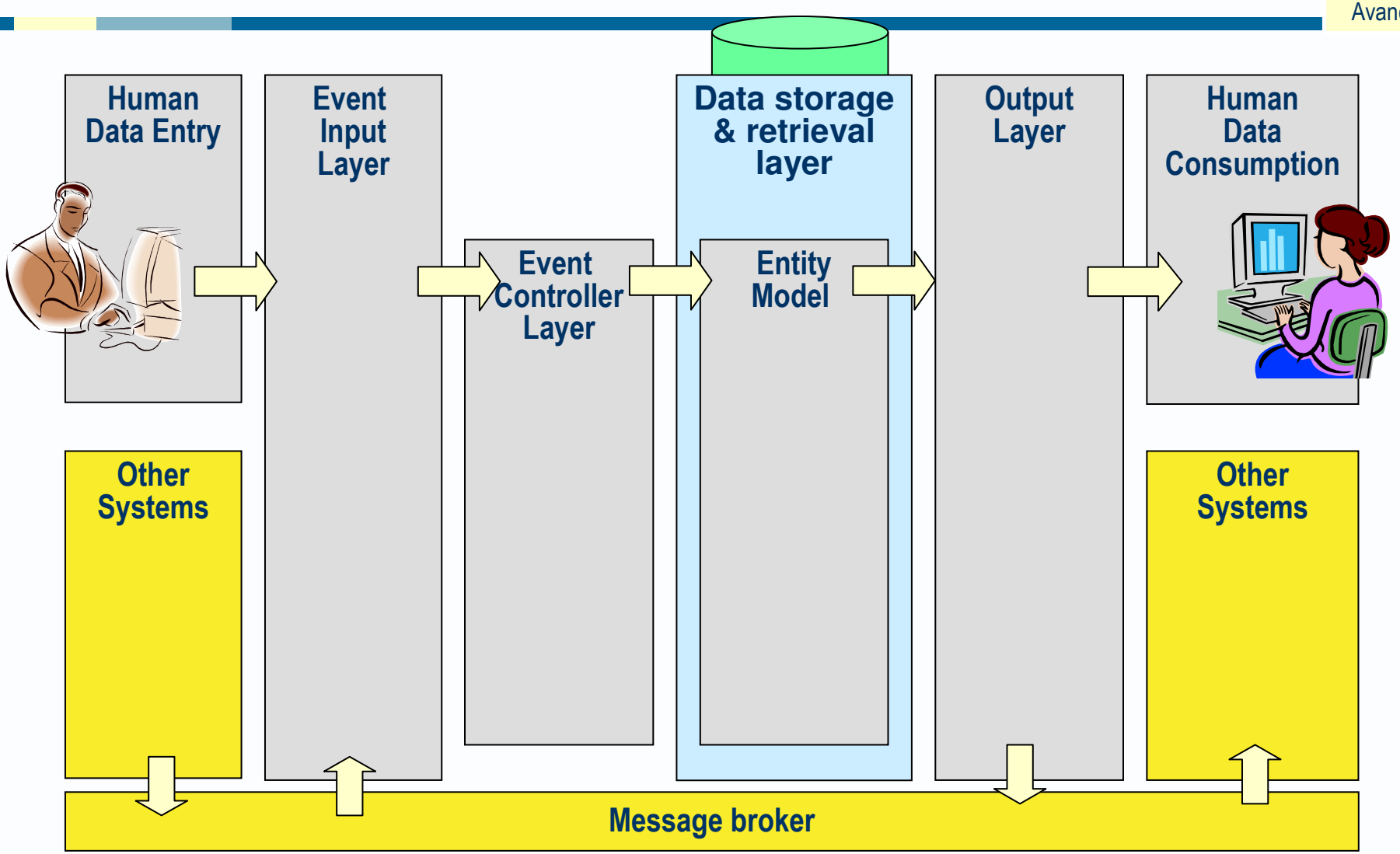
- All the code runs on one computer.

Multi-user client-server system

- Processing is distributed between many computers
- Increasingly, message brokers are used to carry messages between distributed subsystems

# A layer to decouple distributed layers and components

Avancier



# Size and Complexity


Single-user microwaver

- The system is very small; the very few entities can be coded in a single program by one programmer.

Multi-user client-server system

- Often grow very large and complex
  - Many interfaces to external entities
  - Many instances of a class
  - Complex birth and death events
  - Maintenance involves a complex social organisation

# Many instances of a class


Single-user microwaver

- There is only one instance of each class: one door, one light and so on.
- We don't distinguish between class and object
- We don't distinguish between class-level and instance-level 'methods' or 'functions'.

Multi-user client-server system

- tends to features
  - many instances of a class
  - one-to-many associations between classes
  - loops in the interactions diagrams.

# Complex birth and death events

Avancier

Single-user microwaver

- no death events
  - the objects are never destroyed and
- trivial birth
  - we simply coded into each class a constructor method to initialise the object's state variable when the Oven is plugged in and the system becomes active.

Multi-user client-server system

- features events that create and destroy entity object instances.
- The death of a persistent entity object is often constrained by complex business rules and leads to extensive processing

# Maintenance involves a complex social organisation


Single-user microwaver

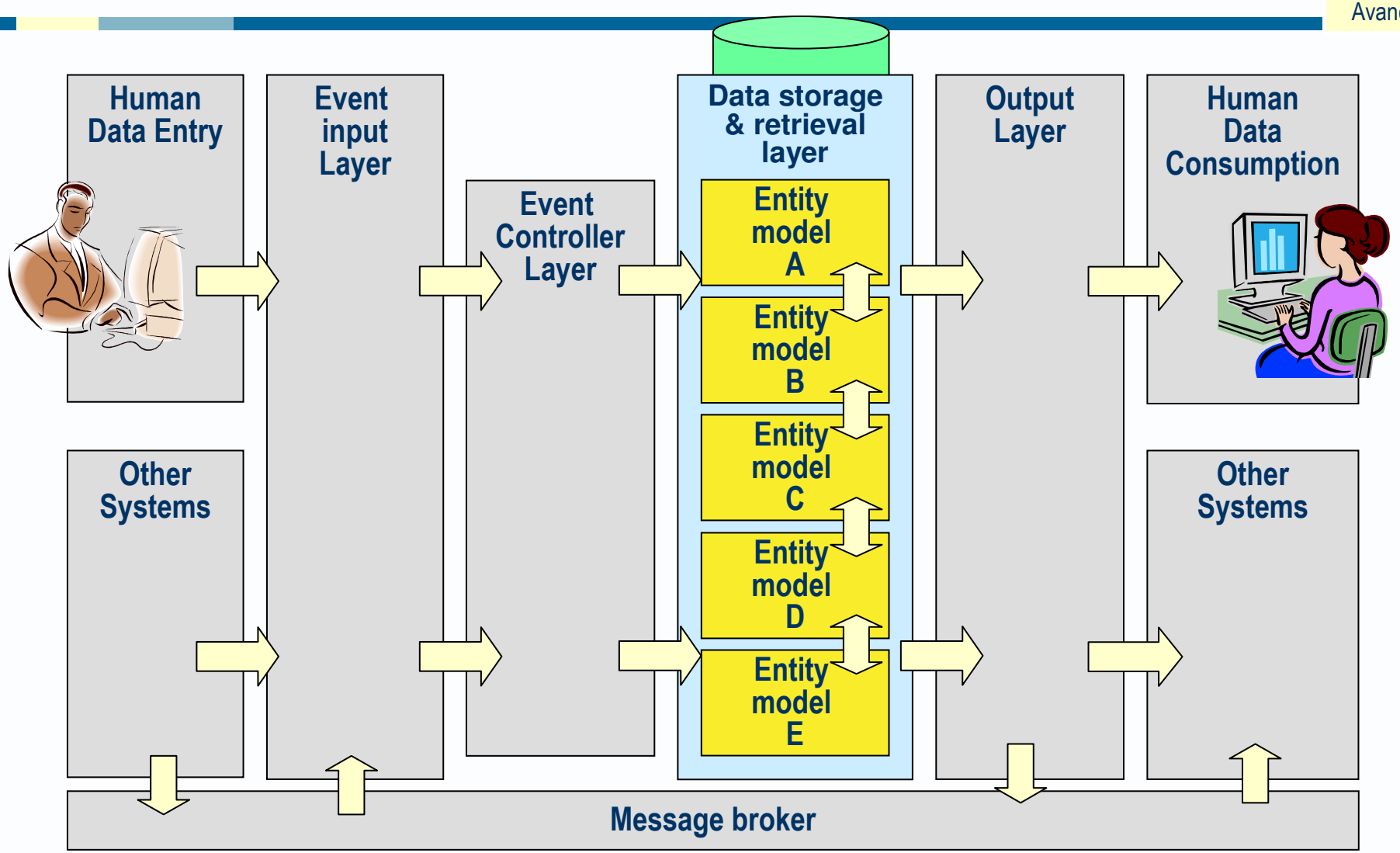
- The system is very small
- The very few entities can be coded in a single program by one programmer.

Large and complex models

- Entity maintenance must be divided between subsystems, just to make the system maintainable by distinct teams
- Yet the discrete entity models must also be synchronised

# The resulting system block diagram

Avancier



# For on-line transaction processing (OLTP)

Avancier