A walk on the wild side

Wednesday, April 20, 2011
2:08 PM

So far, we've studied things that look awfully much like programs.

- AppEngine: entirely like a program
- BPEL: like a flowchart.
- BPMN: like an extension of a flowchart.

Today, we take a walk further out on a limb.

- Force.com: a completely different way to model processes.
- Claim: no coding!

Summary for today

Wednesday, April 14, 2010
2:11 PM

Summary for today

I managed to **torture** the Free version of Force.com into doing something useful, and…

The feeling is **mutual**, I'm sure…

Two "free" accounts

Wednesday, April 14, 2010
2:28 PM

Two "free" accounts:

Free force.com allows data modeling and deployment, but not development of custom functions. You must purchase a license to be able to define your own custom functions. Target: Managers and CTOs.

Free developer.force.com allows development, but not deployment! You must purchase a license to deploy your application. Target: developers.

SalesForce.com

A leading vendor of cloud-based business apps.

Based upon a completely different modeling process than BPMN (and a much older one).

Key to SalesForce.com: Event-Condition-Action (ECA) modeling.

Roots of force.com

- Originally, business process was defined through databases.
- Process consisted of
  - transforms of data
  - lifecycles of a particular record (ECA).
  - "triggers": things that happen when a record is changed.
- In force.com
  - no real databases
  - objects "like" JDOs
  - query language SOQL: "like" JDO queries.
  - difference: define transformations of objects, not flow of actions.

A not-so-subtle irony

BPMN expresses **process flow**, doesn't include details of state (variables) or conditions (logical predicates).

ECA models **state and conditions**, doesn't include details of flow (precedence and messages).

What ECA specifies is almost precisely what BPMN leaves out, that must be re-inserted to create executable BPEL!

ECA modeling

ECA: Event-Condition-Action

A **state machine model** of business

Event: something that happens

Condition: some logical condition (that is a function of state).

Action: some modification of state as a result.

In Force, ECA applies to individual objects, and "wraps" objects in a state machine.

Which implies that objects get big, and are hard to break into pieces.

Developing backwards

- In Force.com, you define data and permissible actions first, then events and conditions.
  - First step: what people "can do" with the system.
  - Second step: what people "should do".
- In BPMN, you define events and conditions first, then define data and actions from them.
  - First step: what people "should do" (without the "can" part).
- Almost precisely backward from what we've learned about modeling so far.

ECA modeling

Begin with an object model of **business data**.

Define **profiles, roles,** and **users** that describe what data state transitions **can** happen:

user: an entity who can modify data.

profile: a set of access privileges for data.

what someone "can" do.

role: a business role, mapped to profiles by inheritance.

what someone "should" do.

Then define **workflows** that describe what data state transitions **should** happen.

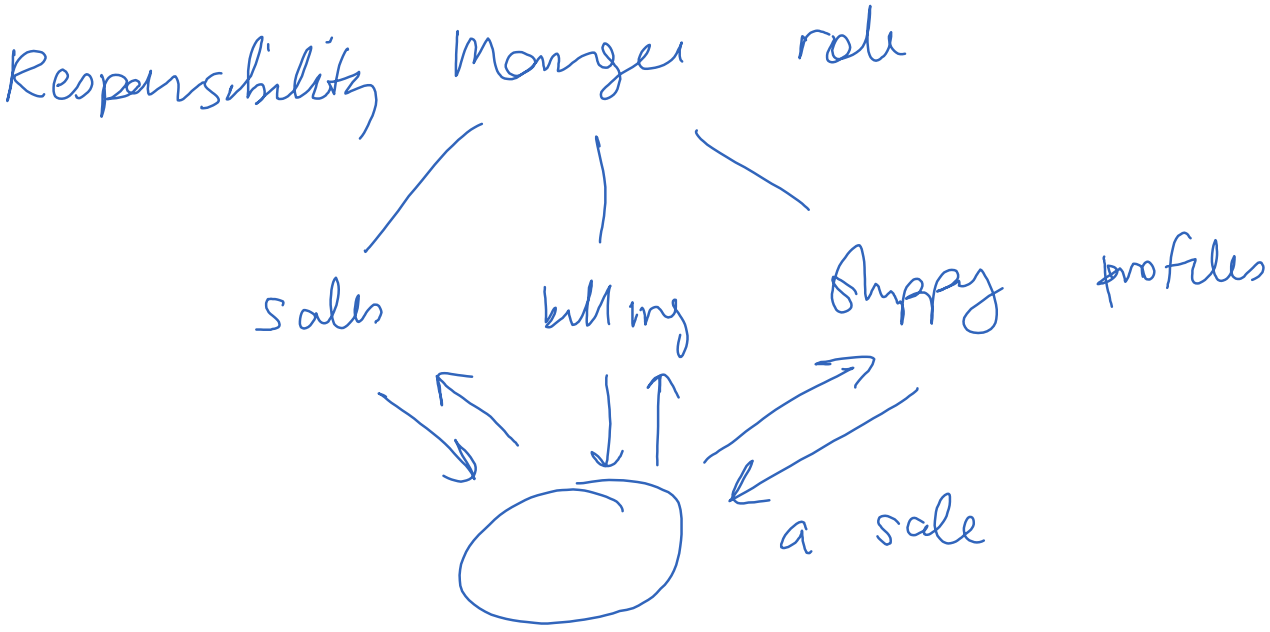
Profiles, Roles, and Users

A **Profile** is a set of user capabilities, e.g., "ability to modify this or that object"

A **Role** describes management hierarchy. Managers inherit the profiles of the people they manage.

Each **User** is assigned a (data access) **Profile** and (optionally) a (business) **Role**.

Wednesday, April 20, 2011
4:44 PM



Wednesday, April 14, 2010
2:36 PM

Roles, management, and inheritance

Profiles describe what users can do.

Roles describe access privileges that (for the most part) arise from managing another person.

A manager **inherits** the profile privileges of the people she manages.

An example of ECA design

Wednesday, April 14, 2010
3:09 PM

An example of ECA design

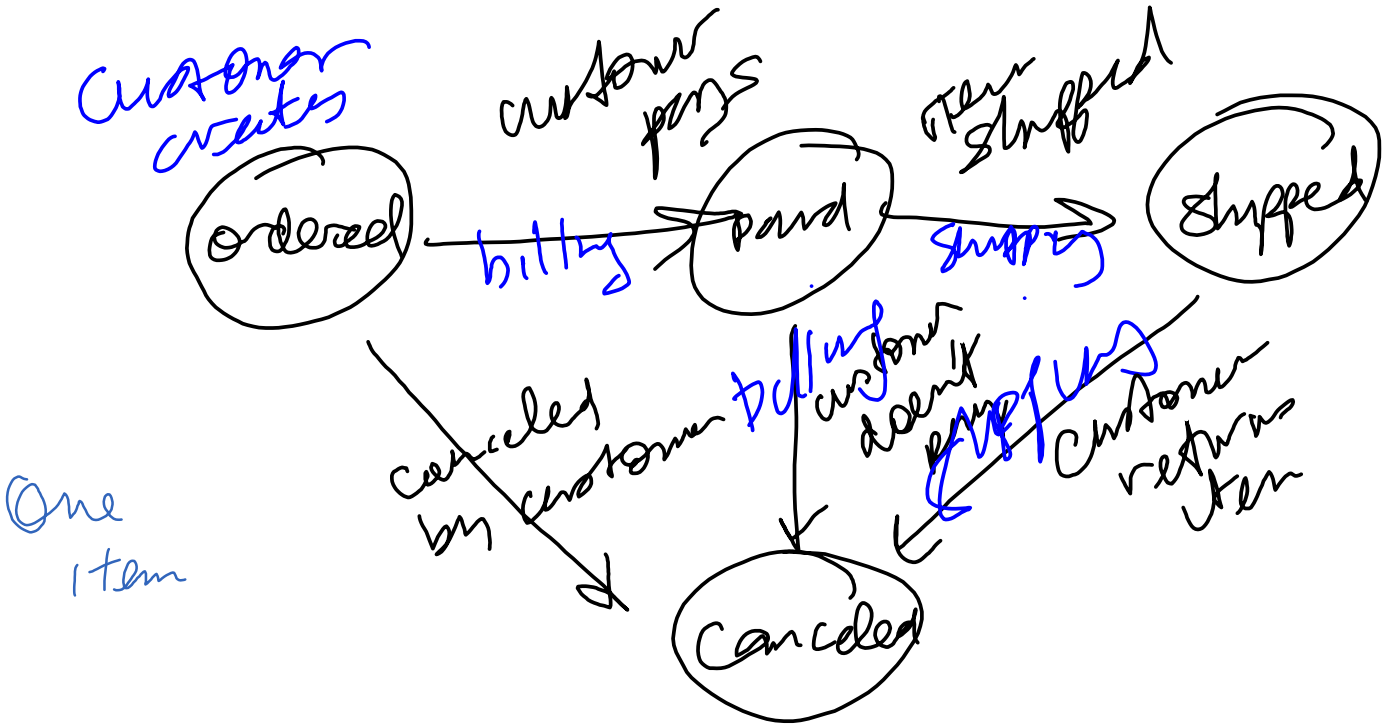
- Case: purchase of an item
- Three entities: customer, billing department, shipping department.

An item can be

- ordered (but not paid)
- paid (but not shipped)
- shipped
- cancelled

An ECA model of process

Wednesday, April 14, 2010
2:59 PM



billing controls "paid"
shipping controls
"shipped" and "canceled"
billing controls "canceled"

Example of data, profiles, and roles

Wednesday, April 14, 2010
2:37 PM

Data, profiles, and roles

- Order record contains **state variables**:
  - **paid**: 'true' if order is paid.
  - **shipped**: 'true' if order is shipped.
  - **returned**: 'true' if order returned.
  - **canceled**: 'true' if order canceled.
- Profiles: **customer**, **billing**, **shipping**
  - **customer** creates Order record, which is partially writeable to **billing** and **shipping**.
  - billing can change 'paid' or 'cancelled' to true in order record, but cannot change the order itself.
  - shipping can change 'shipped' or 'returned' to true in order record, but cannot change whether it is paid.
  - customer can only see **personal** records;
  - billing and shipping can see **all** records.

A strange idea

Wednesday, April 14, 2010
6:24 PM

Note that state is "factored" into variables changed by different principals.
You might ask, why not create a variable "state" that can be "shipped", "paid", etc?
A: then that variable would be writable to all entities, even those that have no "business" writing it.
Sarbanes-Oxley act of 2002: businesses are accountable for enforcing regular processes.

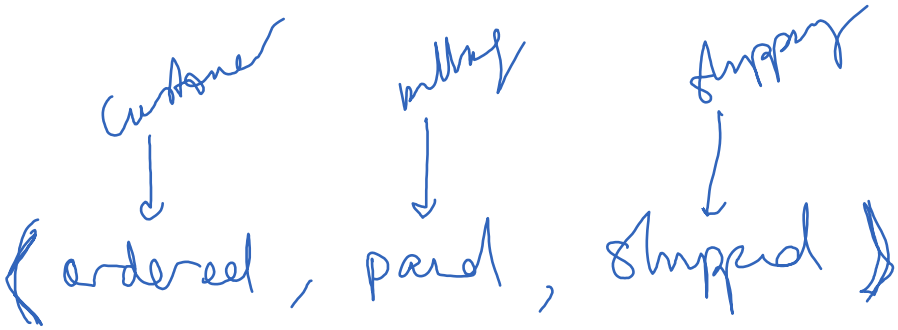
Factoring an object

Decompose the object into a state vector of state components that are writeable to different entities.

Basic principle of factoring: a different role requires a different state component: do not allow two roles access to the same component.

Example: sales roles

Wednesday, April 20, 2011
2:28 PM



Profiles

Wednesday, April 14, 2010
3:08 PM

Profiles: customer, billing, shipping

customer creates Order record, which is partially writeable to billing and shipping.

billing can change 'paid' or 'cancelled' to true in Order record.

shipping can change 'shipped' or 'returned' to true in Order record.

(It is nonsense for customer or billing to set 'shipped' to true!)

customer can only see **personal** records;

billing and shipping can see **all** records.

Implementing ECA in Force.com

Create a data object containing appropriate state.

Set object to be created by customer, and viewed by billing and shipping.

Grant limited rights to modify state to billing and shipping.

Assign profiles to people who are customers, shippers, or billers.

Also need a "manager override" in case things go wrong.

Access rights in force.com

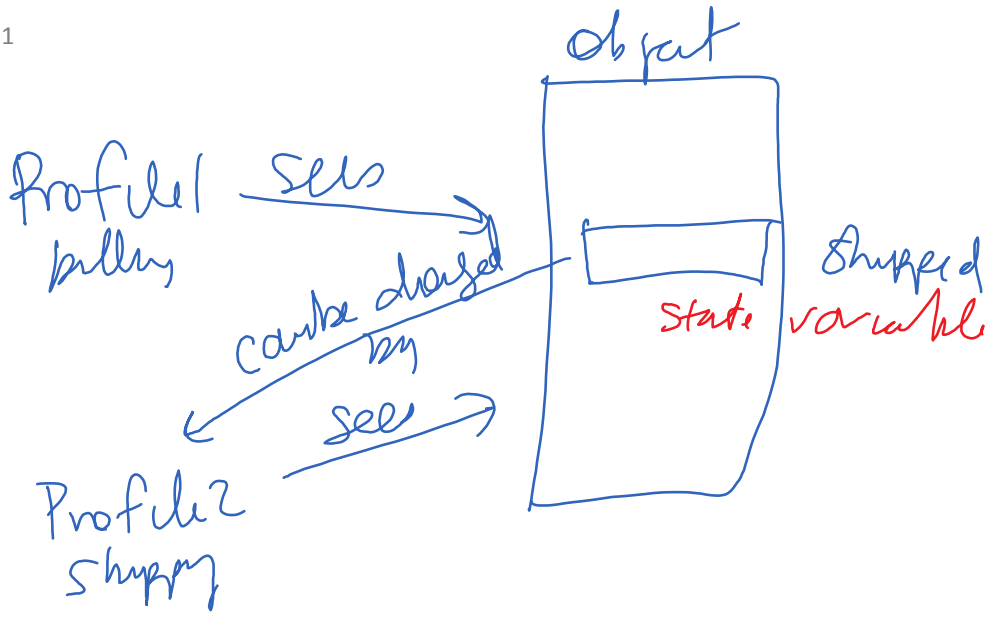
Each profile has overall access rights for each object.

Each object field has access rights for each profile as well.

So, one can make an object viewable, and part of it editable.

Watch out: access rights for objects are properties of the **profile**, while access rights for fields are properties of the object **field** itself, inside the **object**!

Wednesday, April 20, 2011
5:00 PM



Events and actions

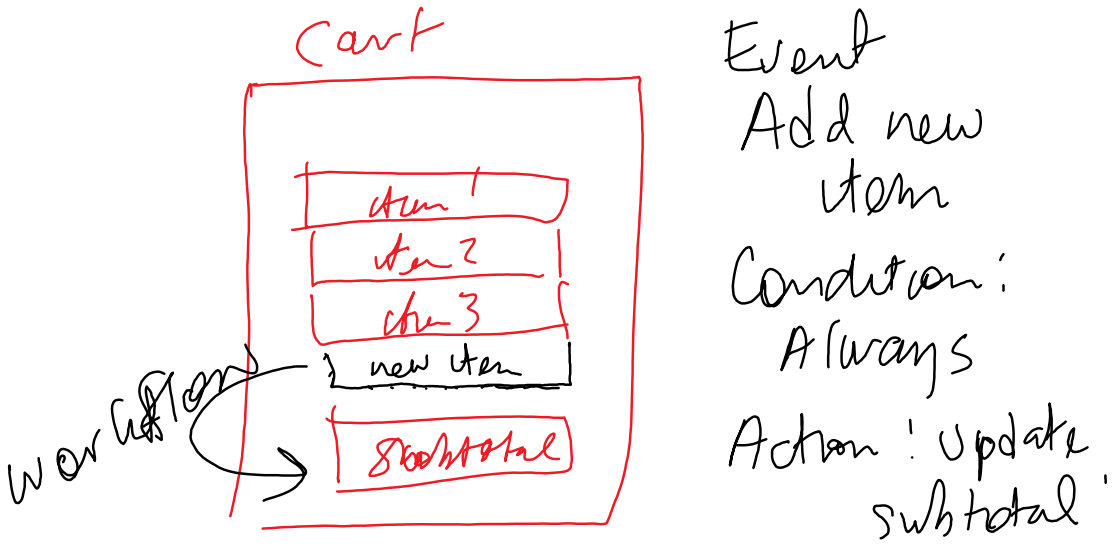
- In Free Force.com, the only events we have are "workflows".
- A "workflow" causes something to happen as a result of a user action, e.g., changing an object record.

Several kinds of workflow actions

- Update field: set a field in an object to a new value based upon other changes in the object.
- Send email.
- Schedule a task (for a user).
- Schedule an "approval" process.

A simple workflow

Wednesday, April 20, 2011
5:05 PM



Understanding "workflows"

Wednesday, April 20, 2011
2:32 PM

A workflow is:

- Roughly the same as a database trigger
- ….but for people!
- If something happens, someone (or some group) in the organization either
  - gets a new task.
  - loses a task.
  - etc.

The Borg Process

Wednesday, April 20, 2011
5:08 PM

"We bring order to chaos"

Tie events to responsibilities.

"If someone buys something, someone else sends a bill."

"If someone pays, someone else ships."

Wednesday, April 14, 2010
4:13 PM

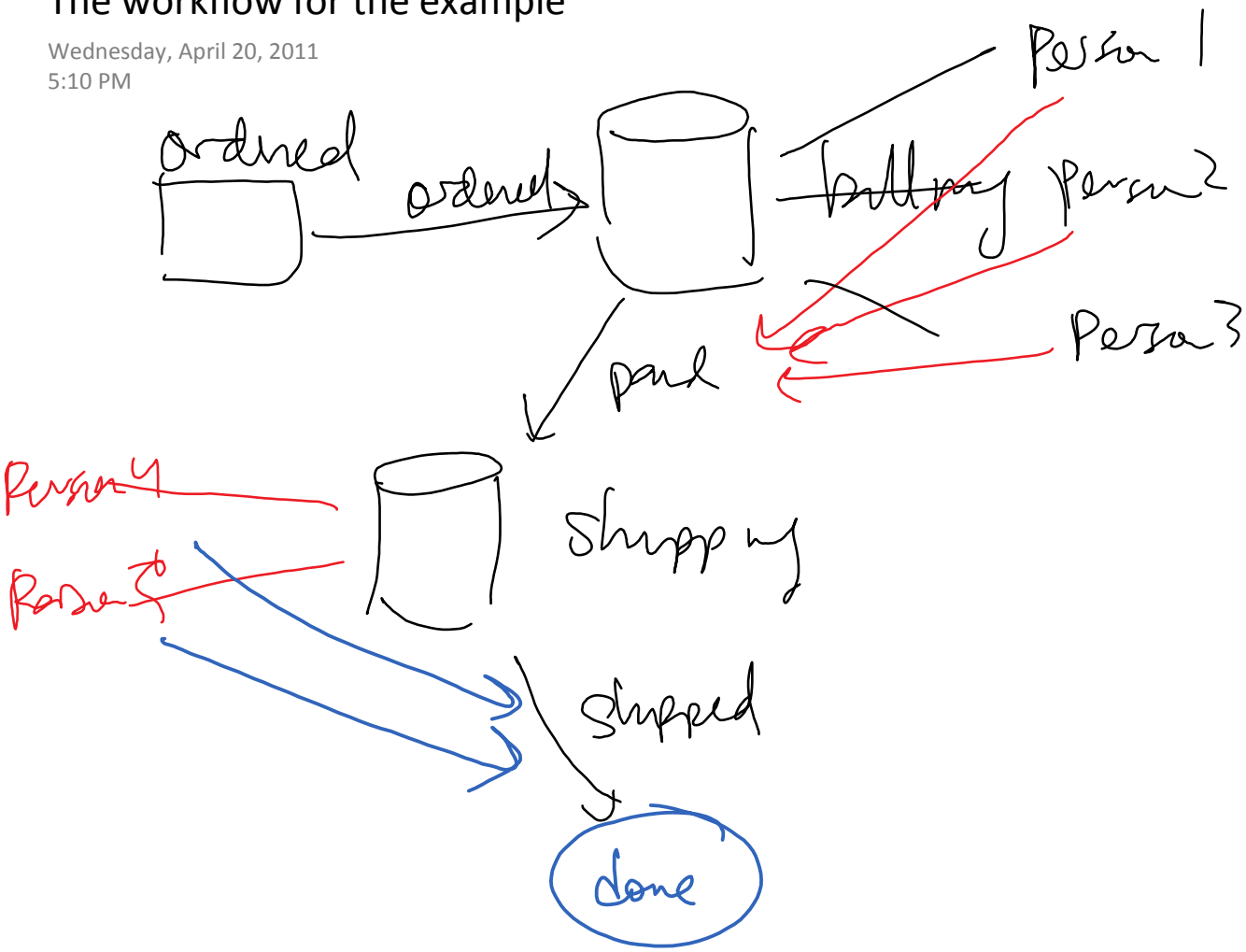
Queues

A queue is a list of tasks that are shared by a group of users (with the same profile).

Can assign a task to either a user or a queue.

The workflow for the example

Wednesday, April 20, 2011
5:10 PM



Approvals

- The most complex feature of Free Force.com: approvals
- Often, business logic requires a complex set of approvals for a record.
  - E.g., for different levels of expenses.
- Approval process requires definition of approvers for data in an object.
  - A potentially complex, multi-state system of approvals is possible.
- Approvals are the only way to "lock" an object in Free force.com.

Approvals are not composite… they cannot be constructed from primitives already available in force.

Force plugins

Wednesday, April 20, 2011
5:18 PM

APEX: extension language for Force
- Not available in free version.
- Purpose: define actions not in the primitive list.
  - Main uses:
    - Complex state transitions
    - Side-effects: link to real world.

Simple example of an extension:
- Plug into inventory system, mark item as present or back-ordered.

More advanced:
- Already a plugin that links Force and AppEngine.