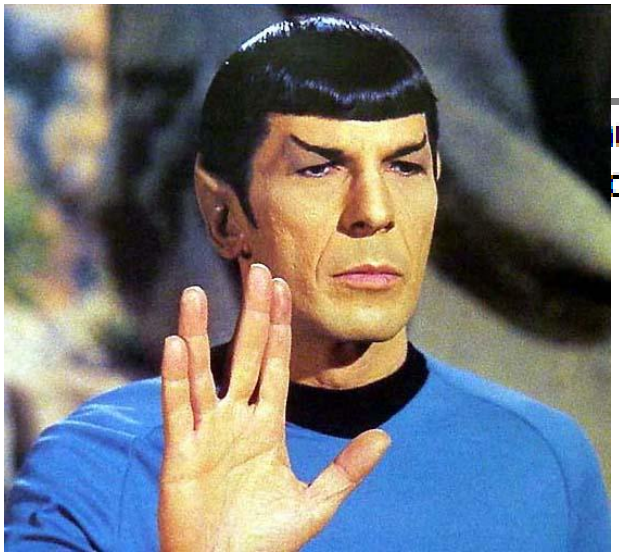

Governor Phillip set aside an open area for the Governor's exclusive use known as the ‘Phillip Domain’. *Despite a ditch being dug to define its boundary in 1792 the Domain was gradually encroached upon by others in subsequent years.* Governor Bligh's attempts to reclaim the Domain was among the many causes of the ‘Rum Rebellion’ of 26 January 1808.
Domain Modeling
Stephen J. Mellor

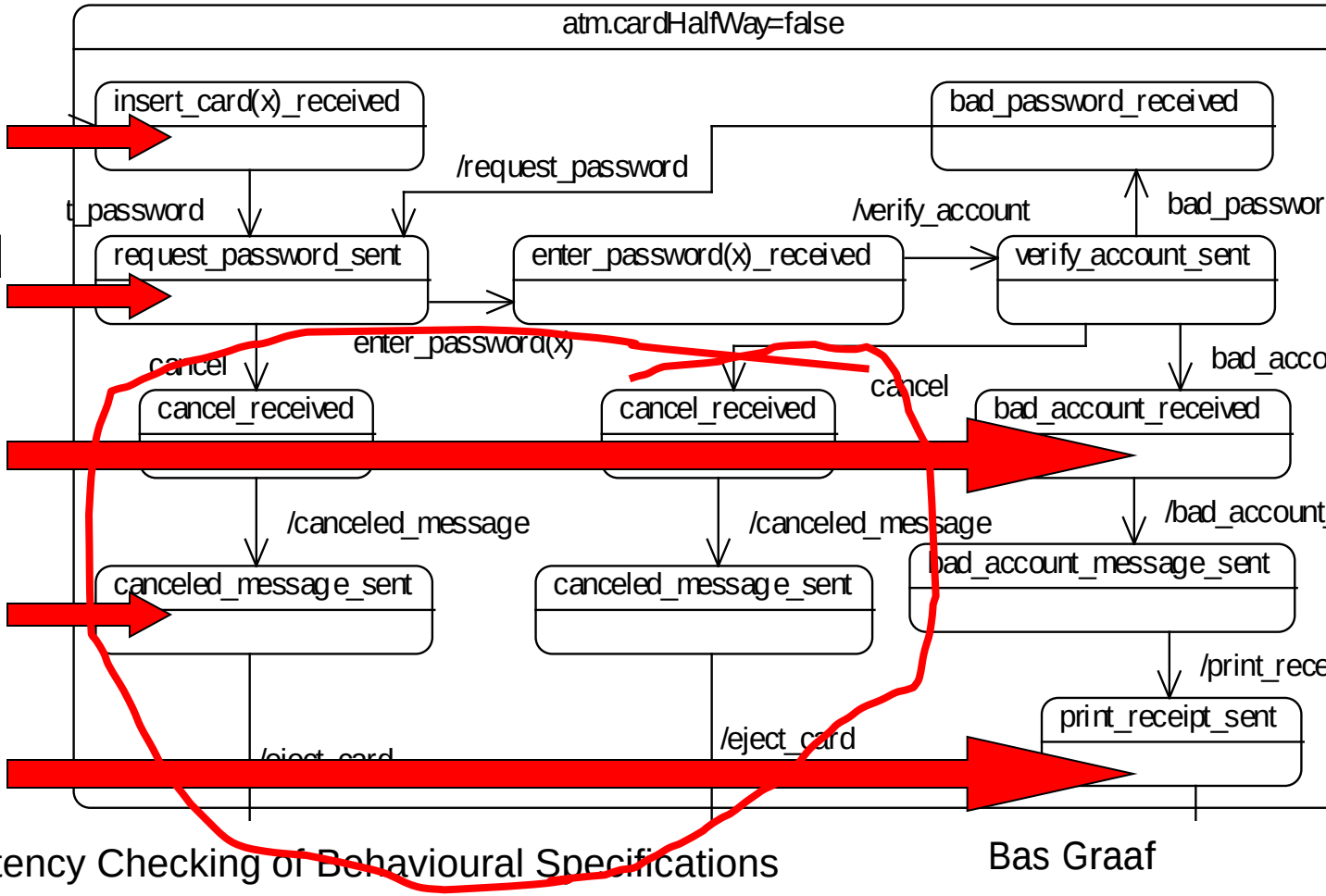
What's Wrong With This Picture?



A UML Documentation for an Elevator System Lu Luo

ATM

- Card
- Password
- Account
- Message
- Receipt!



Model-Driven Consistency Checking of Behavioural Specifications

Bas Graaf

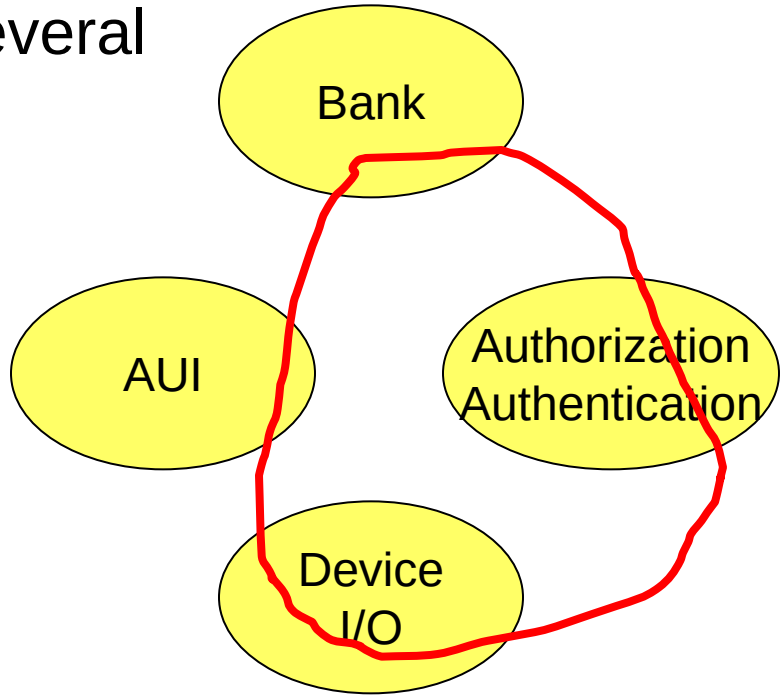
www.laquso.com/VVSS2007/presentations/track10a.ppt

VVSS 2007, Eindhoven

The Bank System

The Bank system comprises several subject matters.

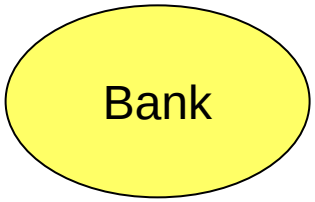
Each subject matter (*domain*) has its own semantics.



Near synonyms include: ontology, conceptual model, problem domain, **aspect**, cross-cutting concern, etc.

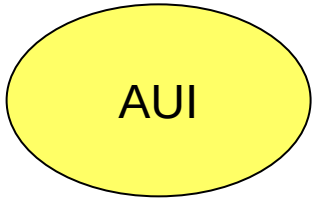
The Bank System

Each domain has its own conceptual entities.



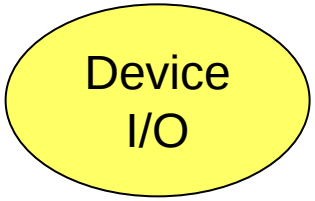
- Customer
- Account
- Transaction

Each domain behaves according to its own rules and policies.

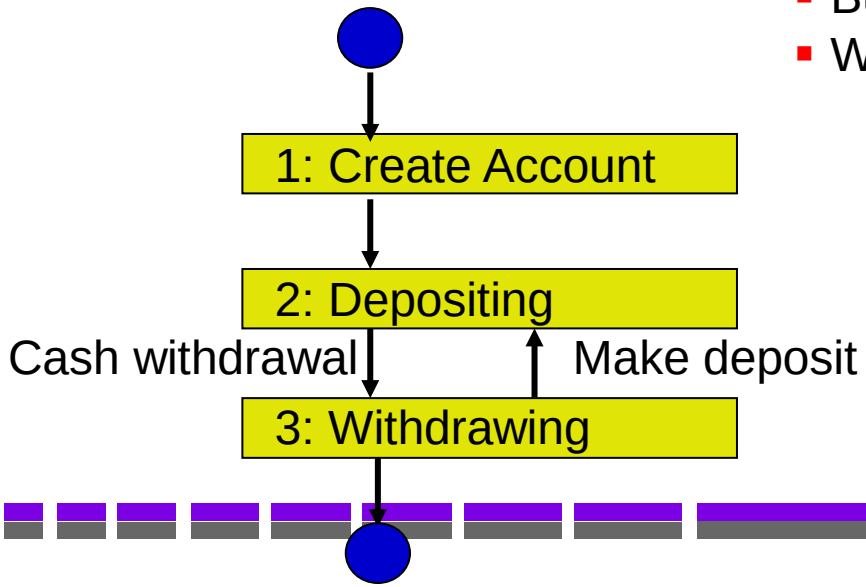


- Button
- Window

- Role
- Protected Resource
- Protected Action
- Permission

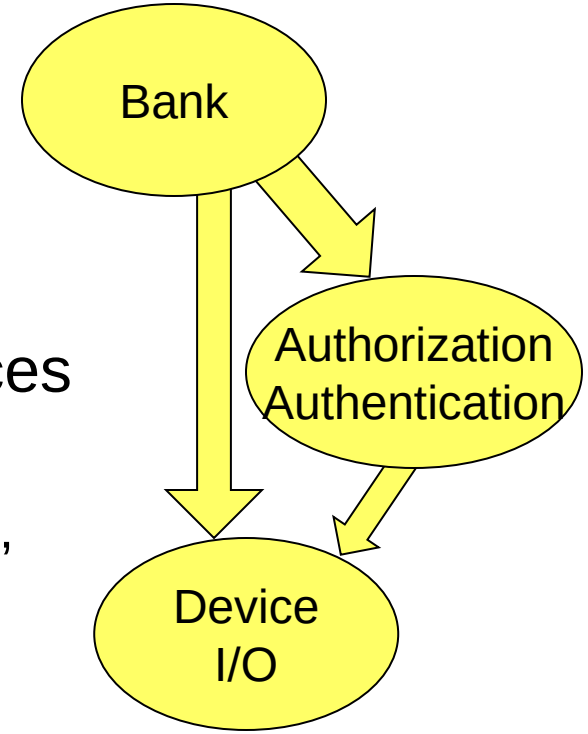


- Serial Bus
- Register I/O
- Command Bitmap
- Keypad



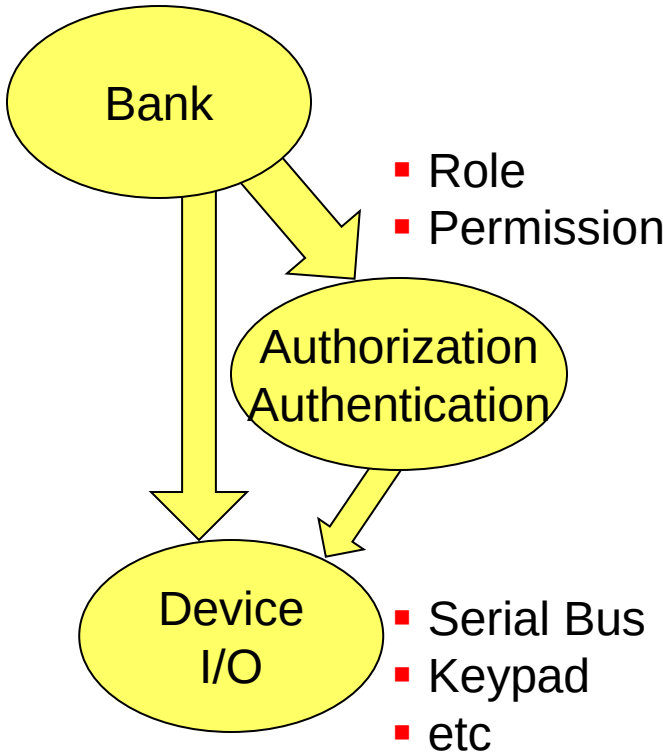
Requirements

- The Bank places requirements on Authorization & Authentication.
  - You must provide me only with valid accounts
- Authorization & Authentication places requirements on Device I/O
  - You must provide me with chip data, entered numbers (*eg PIN*), etc
- The Bank places requirements on Device I/O
  - You must provide me with entered numbers (*eg deposit amount*)
- etc



Services

- Authorization & Authentication provides services to the bank.
  - I provide only roles that have the correct permissions
- Device I/O provides services to both Authorization & Authentication *and* the bank
  - I provide chip data *(eg account data)*
  - I provide numbers from a keypad *(e.g. PIN and deposit amount)*

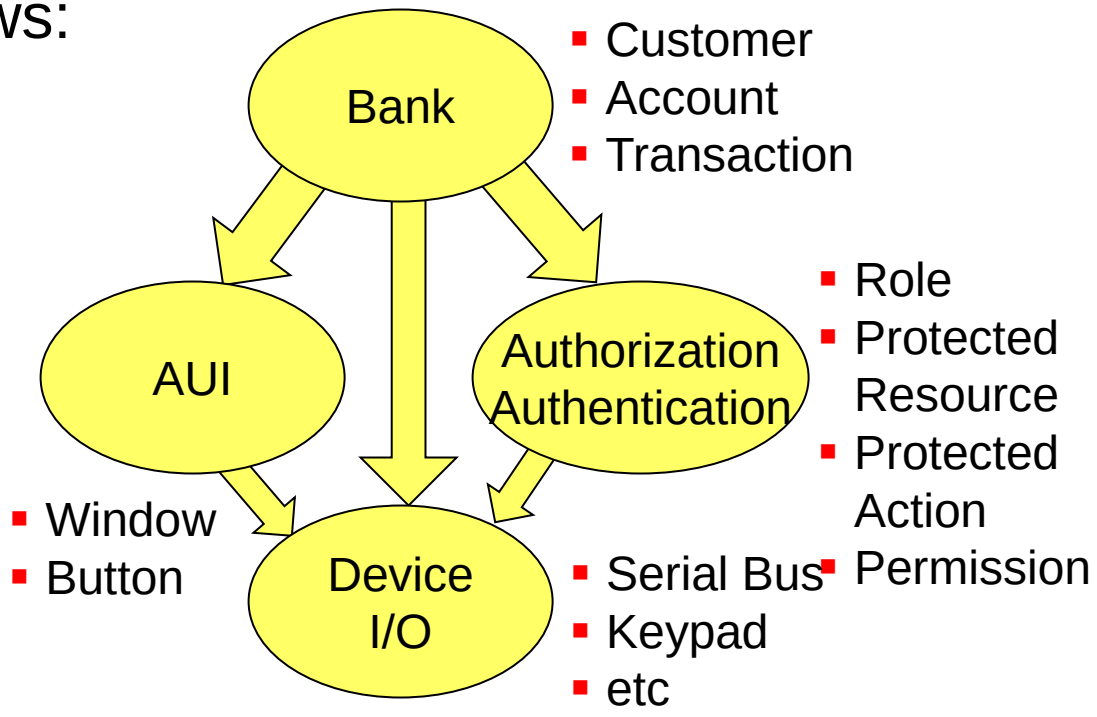


Each domain exports a *meta-interface* to its clients.

Domain Chart

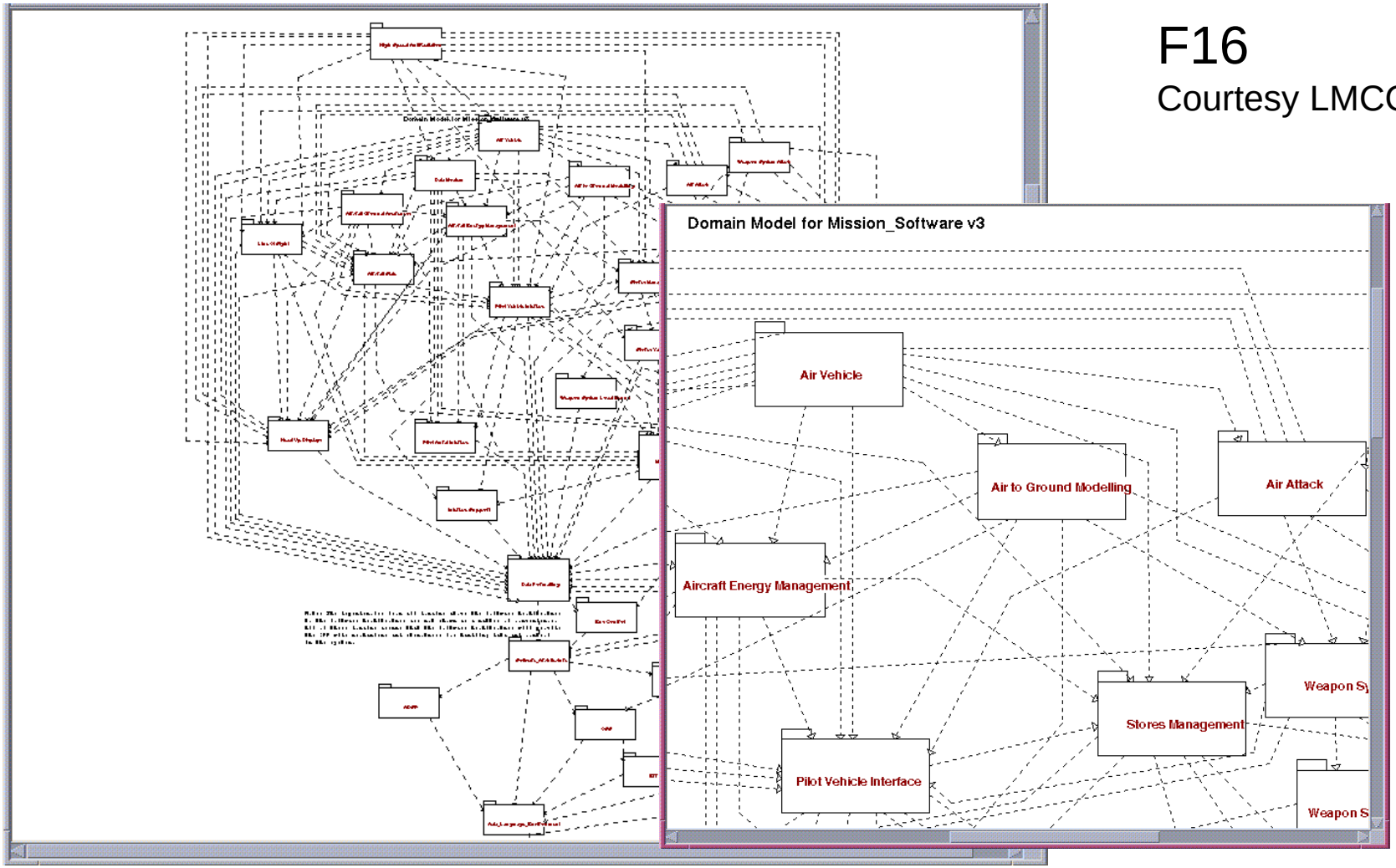
The *domain chart* shows:

- Requirements placed on other domains ("derived requirements")
- Services provided to client domains



The domain chart provides an initial partitioning, and
digs the "ditch" to keep the domains separated.

Big Systems



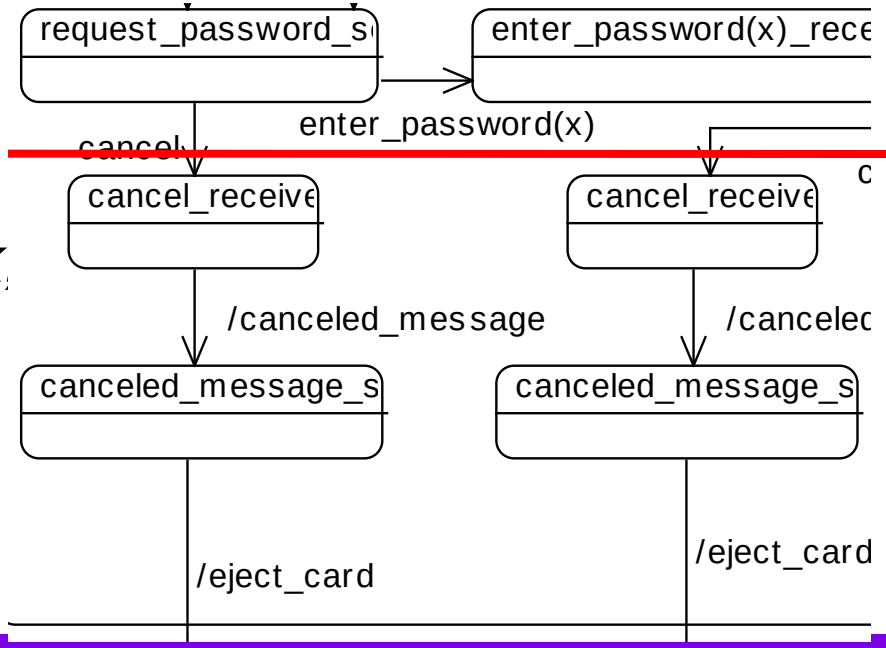
F16

Courtesy LMCO

Why Should I Care?

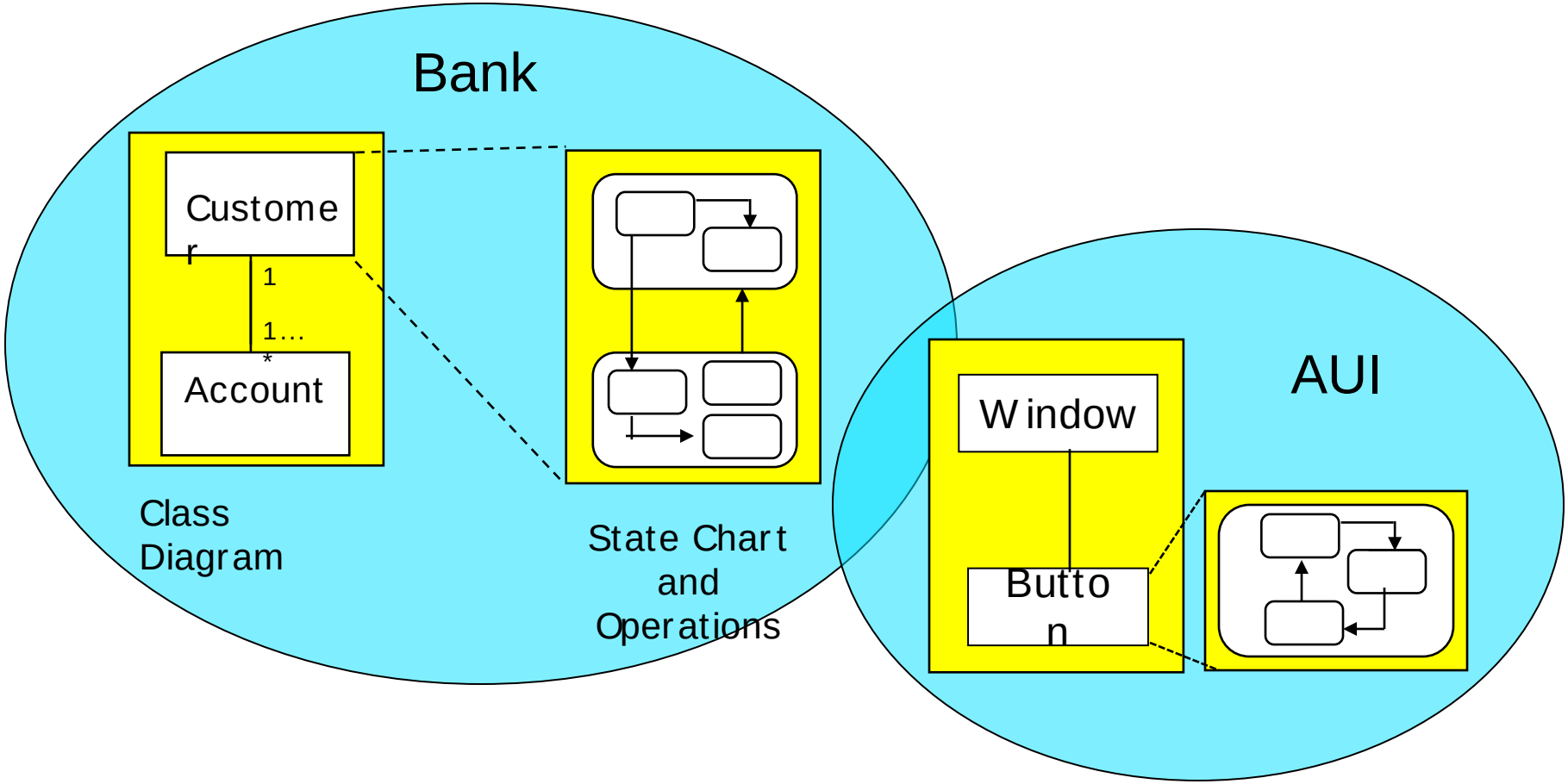
Domain modeling encourages:

- Early identification of separate subject matters
  *(separation of concerns, finding **aspects** ….)*
- Maintenance of that separation
  (it's not a PIN number to the keypad; it is to A&A)
- Work partitioning
  (one teams works the Bank, another the Device I/O)
- Factors the system to reduce redundancy

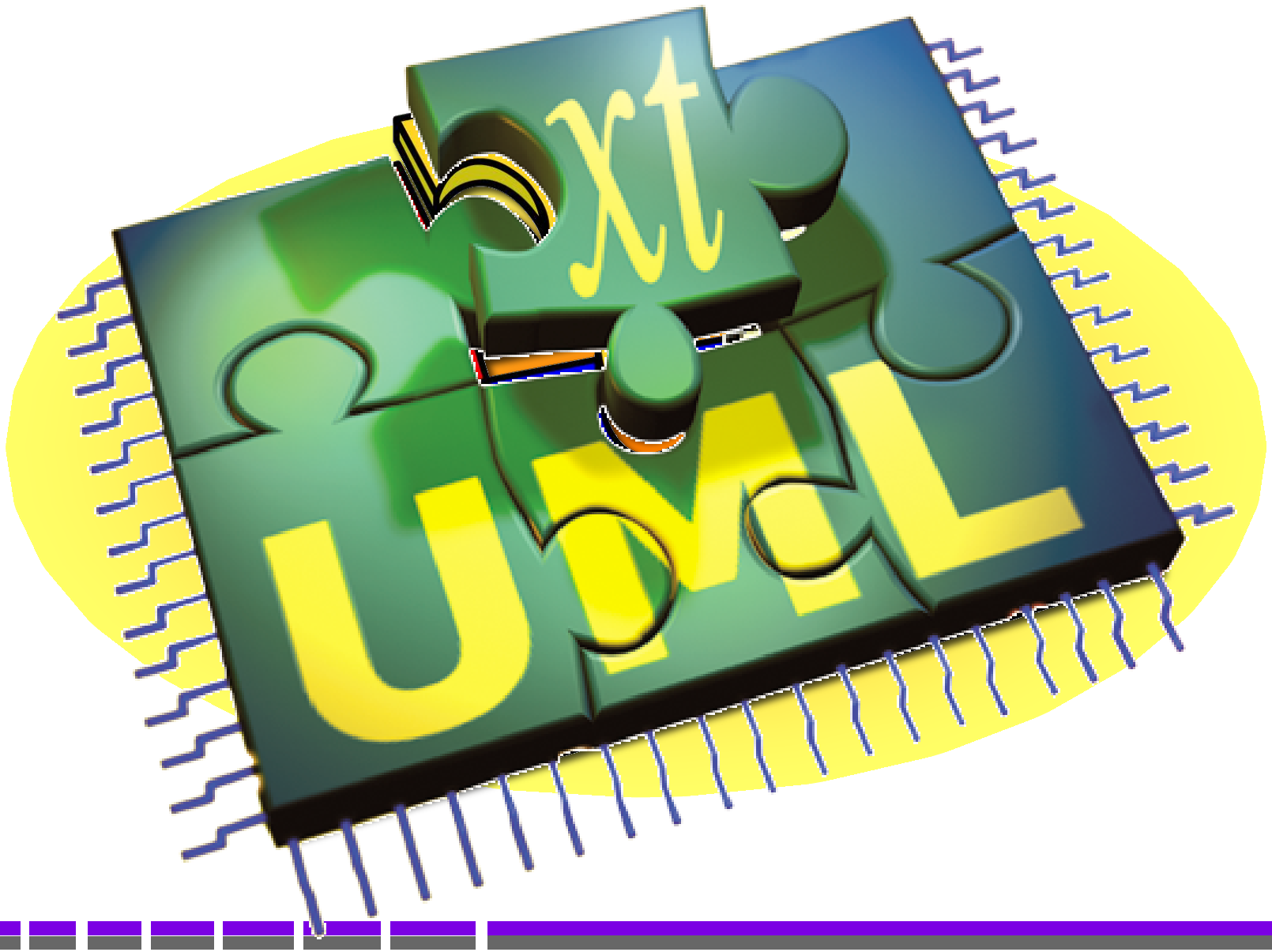


Modeling Domains

Model each domain.

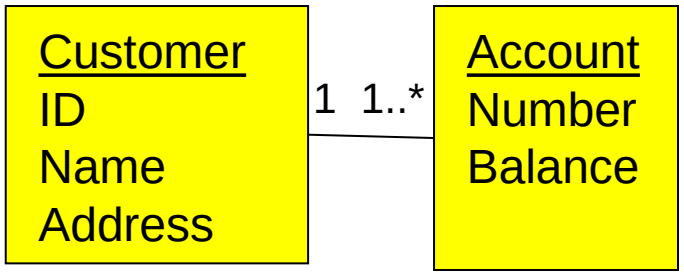


Executable Translatable UML



Metamodels

The bank has a metamodel.

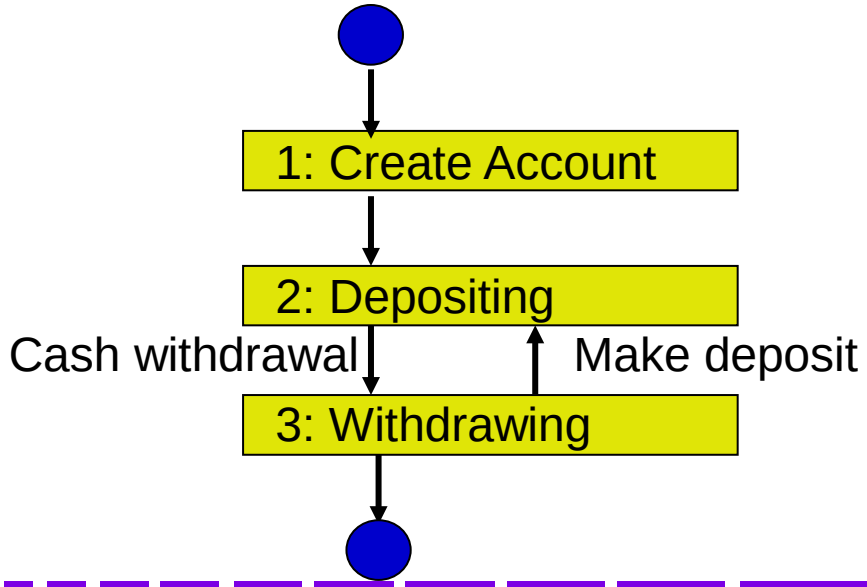


Class

ID	Name
100	Customer
101	Account

Attribute

ClassID	Name
101	Number
101	Balance



Vertex

Class ID	State	Event	New State
Account	2	p	3
Account	3	q	2

Event

Class ID	Event	Name
Account	q	Make deposit
Account	p	Cash withdrawal

Metamodels

So does the AUI.

Window
ID

Button
Nbr
x
y

Class	
100	Window
101	Button

Attribute
Window ID
Button Nbr
Button x
Button y

Vertex

Class ID	State	Event	New State
Button	1	a	2
Button	2	b	2

Event

Class ID	Event	Name
Button	a	Create button
Button	b	Button pressed

Model Database Instances

Each domain has instances during initialization and execution.

Customer
You
Me

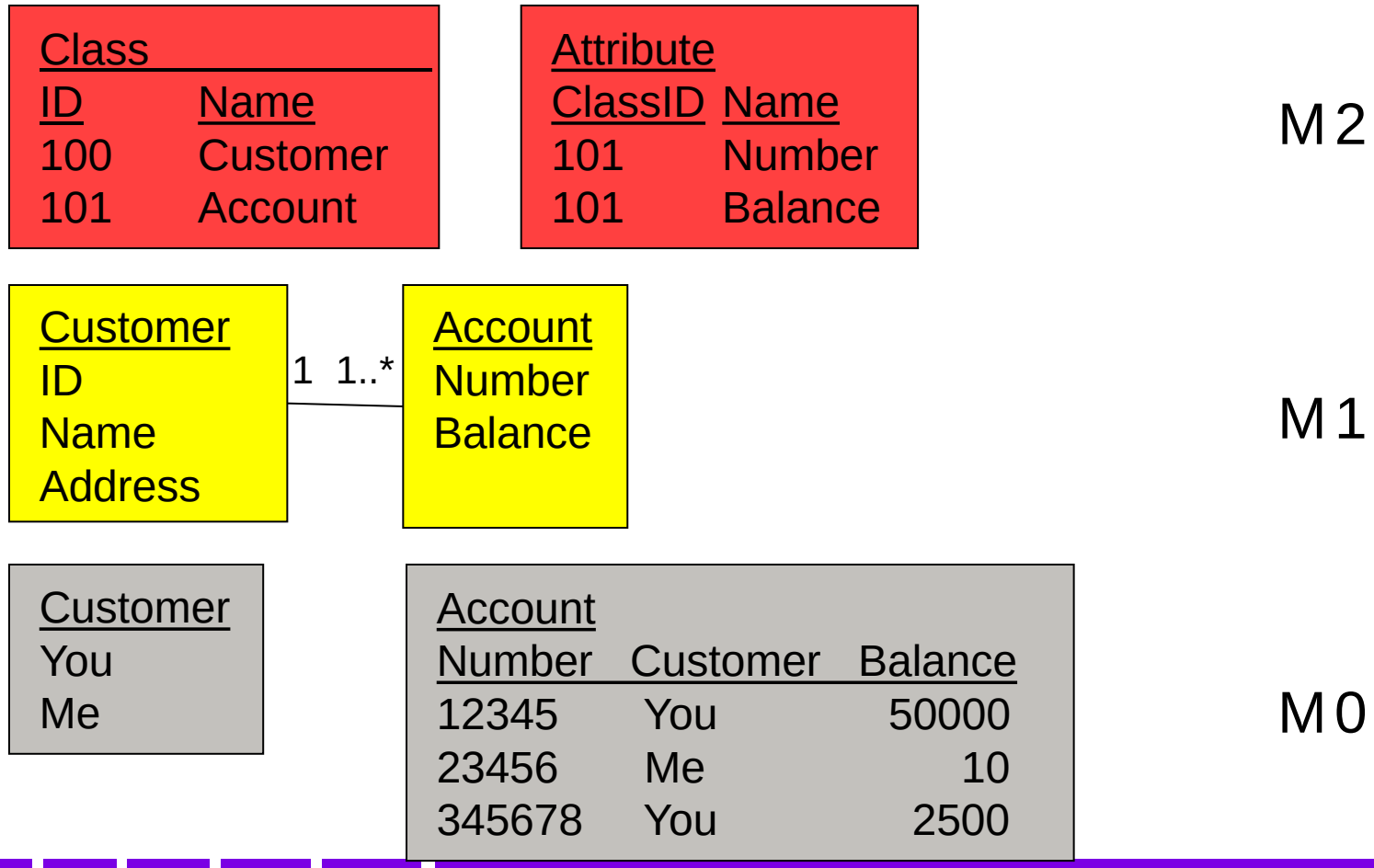
Account		
Number	Customer	Balance
12345	You	50000
23456	Me	10
345678	You	2500

Window
Initial Window
Account Window

Button		
Nbr	x	y
1	12	50
2	12	100
3	12	150

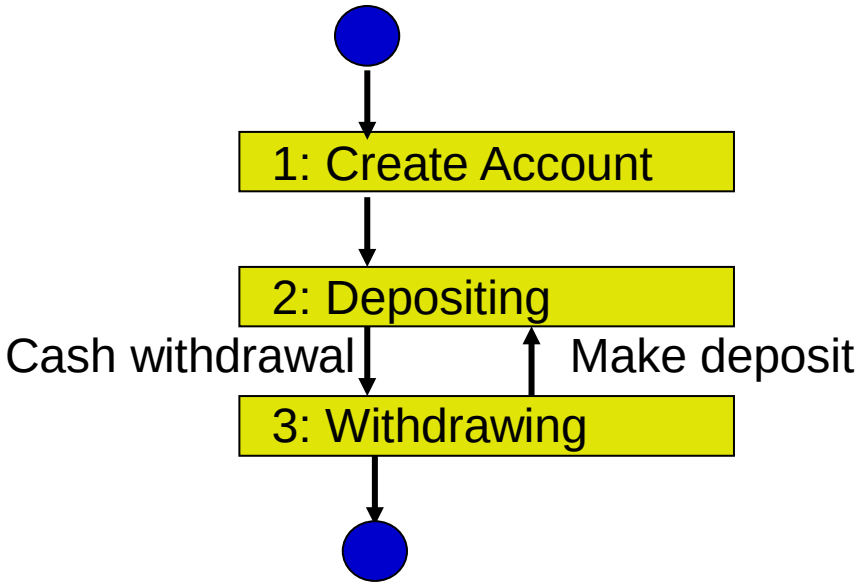
Metalevels

Each domain exists at three levels.



Link Domains

Link the elements in each domain.



Button(1) corresponds to the event Make Deposit.

Button(2) corresponds to the event Cash Withdrawal.

Vertex

Class ID	State	Event	New St
Account	2	p	3
Account	3	q	2

Event

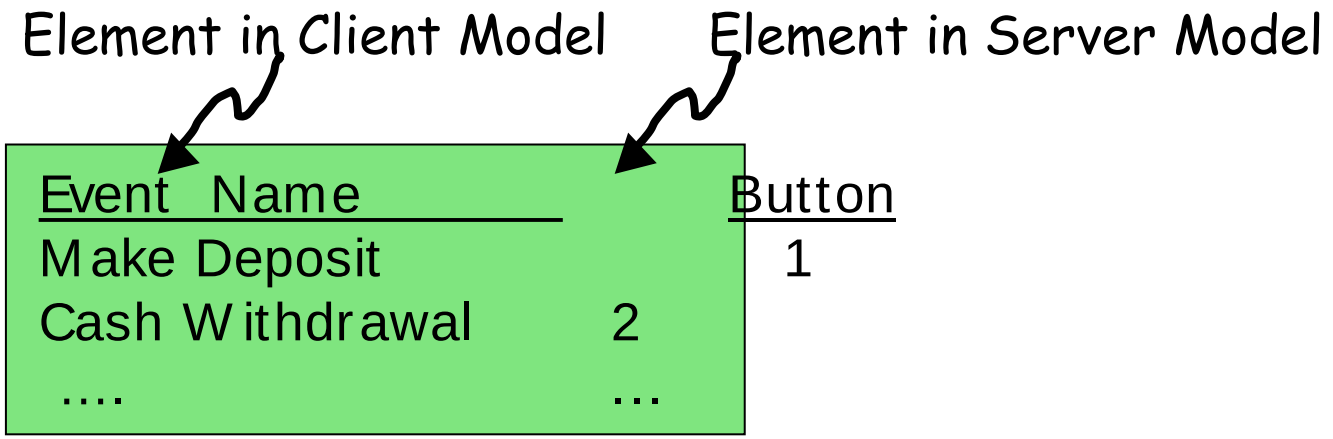
Class ID	Event	Name
Account	p	Make D
Account	q	Cash w

Button

Nbr	x	y
1	12	50
2	12	100
3	12	150

Joining Domains

Domains may be joined by tables…



Or by rules…

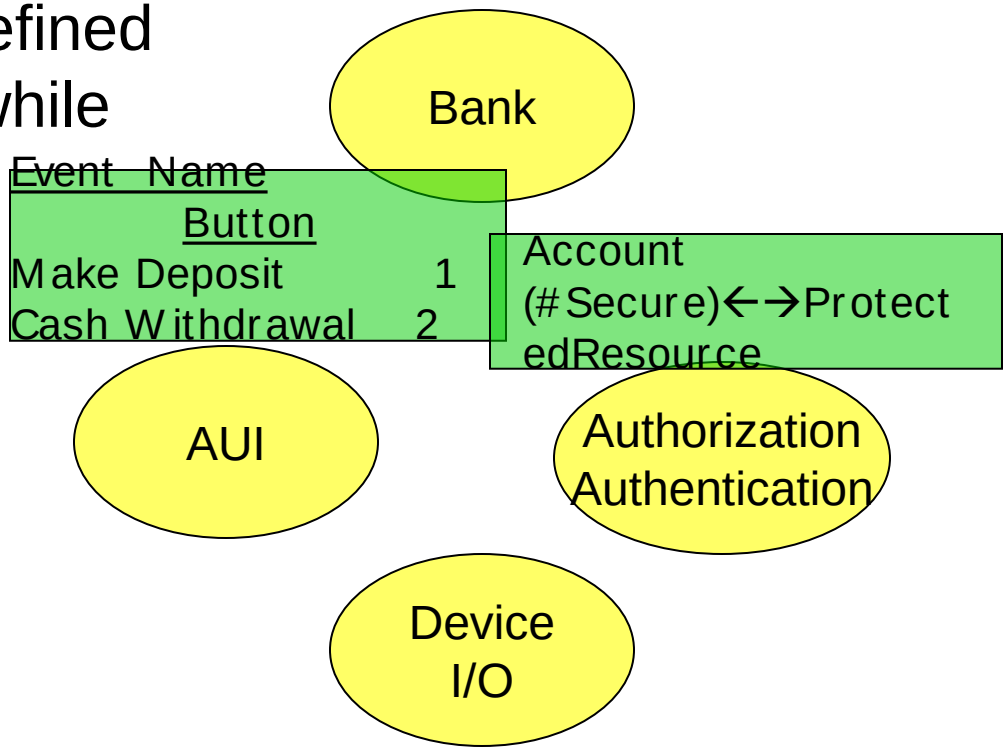
All (Bank) classes marked #Secure correspond to an (A&A) ProtectedResource.

(eg Account (#Secure)←→ProtectedResource)

Why Do I Care?

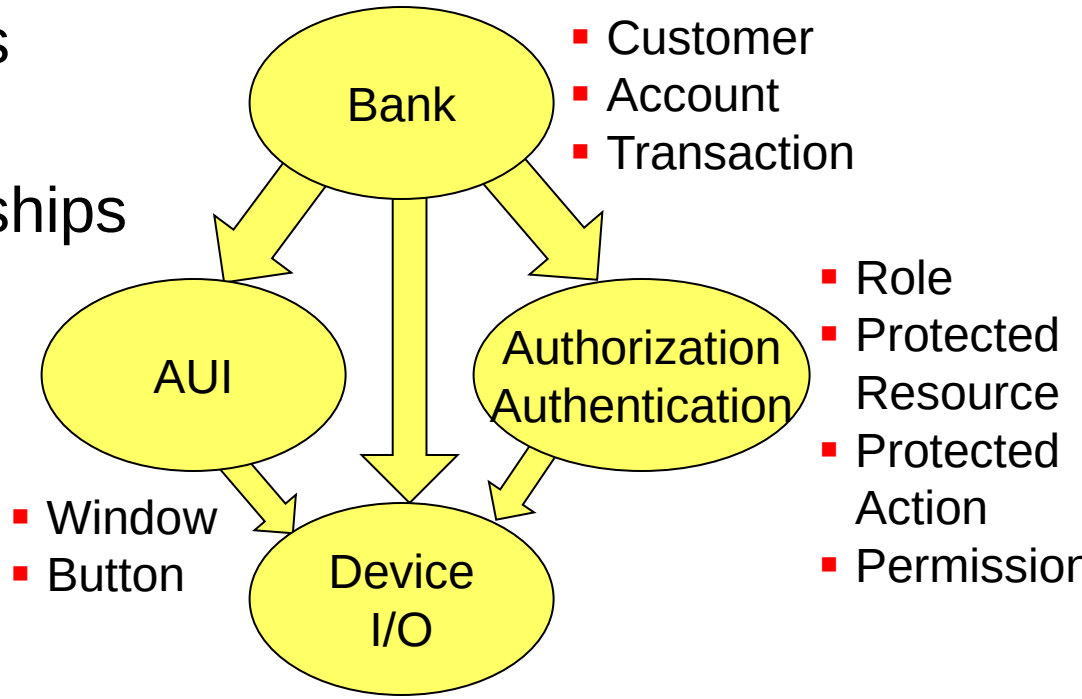
- Each domain is *entirely* separate from others
- Domains can therefore be *reused intact*

- Meta-interfaces are defined *early* in the process, while the actual interface is delayed
- Interfaces can be generated to order



Domain Modeling

- Identify the domains (aspects)
- Identify the relationships between them (services)
- Model each domain
- Link them together



Benefits of Domain Modeling

Domain modeling:

- Provides a mechanism for identifying aspects early
- Provides, at system-definition time, a way of allocating and finding requirements
- Keeps aspects separate during the project
- Partitions the work
- Provides, at system-construction time, a means for linking subject matters together
- Maximizes reuse

Thank you

StephenMellor@StephenMellor.com