Techniques for Increased Action Diagram Logic Reusability

Session 380

Greg Moll
Texas Instruments

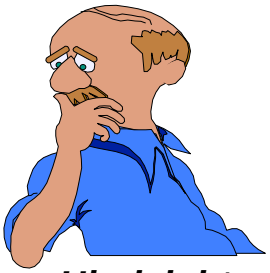
Agenda

Date/Time

Persistent Views

USERID

Wrapping External Action Block



Hindsight

Exit States

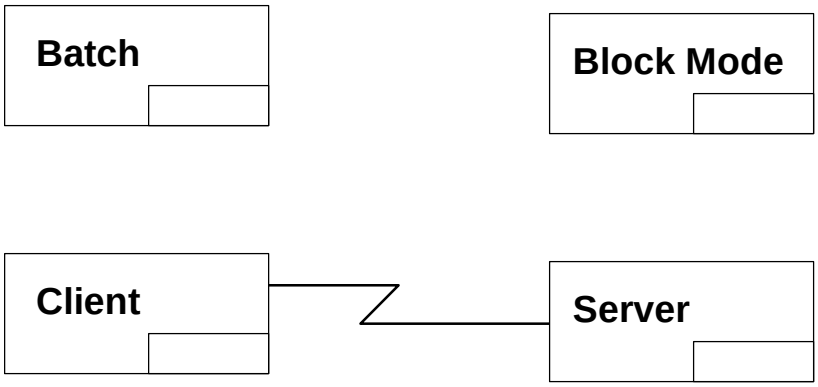
Errors

Session Objectives

- Increase awareness of reusability techniques
- Extend software quality
- Increase maintainability
- Improve developer productivity

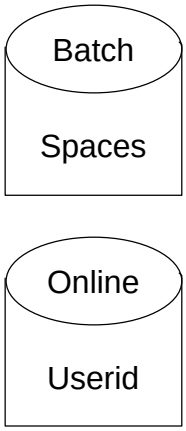
Goal...

Be prepared to reuse your action blocks across architectures

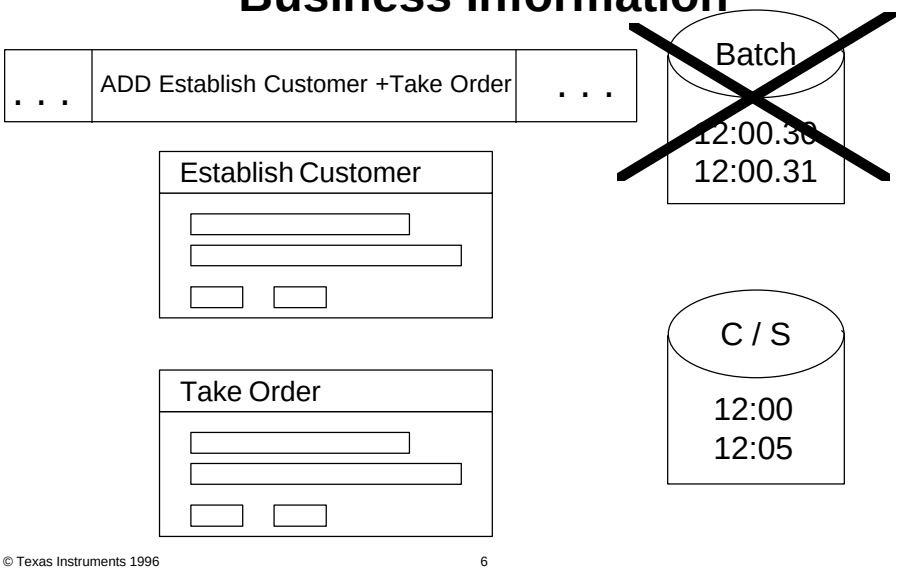


Situation: Incomplete Business Information

```
+- Establish Customer
|   IMPORTS:...
|   EXPORTS:
|   LOCALS:
|   ENTITY ACTIONS:
|     Entity View customer
|       ...
|       userid
|
|  +- CREATE customer
|  |  ...
|  |  SET userid TO USER_ID
|  +- WHEN successful
|  +- WHEN already exists
|  +- WHEN permitted value violation
|  +--
+--
```



Situation: Inconsistent Business Information



Import Special Attributes

- Import Special Attributes into Action Blocks instead of using:
  - Set view XXX to “Function”
- USERID, Date, Time, Timestamp
- Moves control to a higher level
- Overcomes platform and design problems
  - e.g., Timestamp precision
- Improves performance by reducing redundancy

Situation: To be or not to be Persistent

Persistent

Not Persistent

Persistent Views

- Reference stored data
- Support entity actions
  - Actions are allowed on a persistent view:
    - »CREATE, READ, UPDATE, DELETE, ASSOCIATE, DISASSOCIATE, and TRANSFER
- Must be the exact same attributes on the entity views being matched
- Matching between called or calling *action block* view must be persistent or from an Entity Action View

Locked

- Indicates whether the generated SQL for a READ in a calling action block should include an “intent to update” phrase
- Only needed if passing currency on a persistent view

Parent to Child

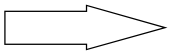
- Read in one action block enables the following in a ***called*** action block:
  - If persistent and both imported and exported:
    - »Delete, Update
    - »Associate, disassociate, transfer
    - »Where CURRENT clause for a READ or READ EACH
  - If persistent:
    - »Where CURRENT clause for a READ or READ EACH

Parent to Child Usage

Parent Action Block

```
READ x

USE Child
```



Child Action Block

```
UPDATE x

CREATE y
  ASSOCIATE WITH x

READ z
  WHERE DESIRED z is
  related_to CURRENT x
```


Child to Parent

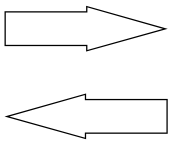
- Can perform an update in called action block and then do the following in the ***calling*** action block:
  - Where CURRENT clause for a READ or READ EACH

Child to Parent Usage

Parent Action Block

```
USE Child

READ z
  WHERE DESIRED z is
  related_to CURRENT x
```



Child Action Block

```
READ x

UPDATE x
```


Parent to Child Update Example

- Already read Project entity in Parent Action Diagram and want to update it in Child Action Diagram
- Match: Child imp_exp_eav Project to Parent entity action or persistent view
- Caution: Child action diagram should always test that the view Project is populated and locked. If not, it should perform the read or return an error message

Child Views

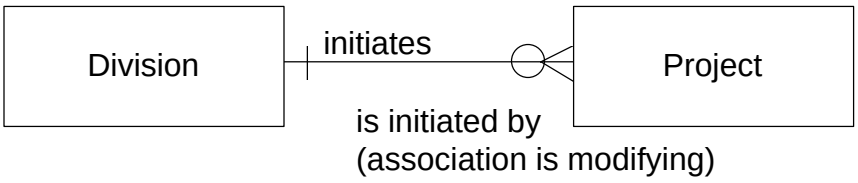
```
IMPORTS:
  imp_exp_eav project (opt, pers, exp, locked)
    number (optional)
    name (optional)
  import project (opt, trans, import only)
    number (optional)
    name (optional)
ENTITY ACTIONS:
```


Child Action Diagram Logic

```
 +- IF imp_exp_eav project IS NOT POPULATED
 |  OR imp_exp_eav project IS NOT LOCKED
 |  +- READ imp_exp_eav project
 |  |    WHERE DESIRED imp_exp_eav project number
 |  |          IS EQUAL TO import project number
 |  +- WHEN SUCCESSFUL
 |  +- WHEN NOT FOUND
<------ESCAPE
 |  +-
 +-

 +- UPDATE imp_exp_eav project
 |  SET name TO import project name
 ...
```


Data Model for Example



Association for Modifying Rel.

- Already read Division entity in Parent Action Diagram(AD) and want to associate it to Project in Child AD
- Match: Child AD imp_exp_eav Division to Parent AD entity action or persistent view
- Caution: Child should always test that the view is populated and locked. If not, it should perform the read or return an error message

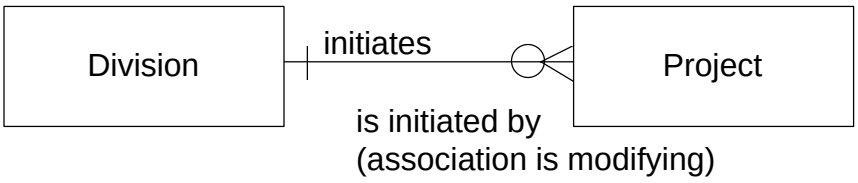
Association for Modifying Rel. (cont.)

```
IMPORTS:
 imp_exp_eav division (opt, pers, exported, locked)
    number (optional)
 import division (opt, trans, import only)
    number (optional)
 import project (mand, trans, import only)
    number (mandatory)
    name (mandatory)
ENTITY ACTION
  new project
    number
    name
```


Association for Modifying Rel. (cont.)

```
 +- IF imp_exp_eav division IS NOT POPULATED
 |  OR imp_exp_eav division IS NOT LOCKED
 |  +- READ imp_exp_eav division
 |  |    WHERE DESIRED imp_exp_eav division number
 |  |          IS EQUAL TO import division number
 |  +- WHEN SUCCESSFUL
 |  +- WHEN NOT FOUND
<------ESCAPE
 |  +-
 +-
 +- CREATE new project
 |  SET ...
 |  ASSOCIATE WITH imp_exp_eav division
 |  WHICH initiates IT
 ...
```


Data Model for Example



Association with Reference

- Already read Division entity in Parent Action Diagram and want to associate to entity Project in Child AD
- Match: Child imp_exp_eav Division to Parent AD entity action or persistent view
- Caution: Child AD should always test that the view is populated. If not, it should perform the read or return an error message

Association with Reference (cont.)

```
IMPORTS:
 imp_exp_eav division (opt, pers, exported)
    number (optional)
 import division (opt, trans, import only)
    number (optional)
 import project (mand, trans, import only)
    number (mandatory)
    name (mandatory)
ENTITY ACTION
  new project
    number
    name
```


Association with Reference (cont.)

```
 +- IF imp_exp_eav division IS NOT POPULATED
 |  +- READ imp_exp_eav division
 |  |    WHERE DESIRED imp_exp_eav division number
 |  |          IS EQUAL TO import division number
 |  +- WHEN SUCCESSFUL
 |  +- WHEN NOT FOUND
<------ESCAPE
 |  +-
 +-

 +- CREATE new project
 |  SET ...
 |  ASSOCIATE WITH imp_exp_eav division
 |  WHICH initiates IT
 ...
```


Example View Naming Standards

- Import (exported)
  - Standard: **imp_exp**
- Import (exported, persistent)
  - Standard: **imp_exp_eav**
  - applies whether view is locked or unlocked
- Export (imported)
  - Standard: **exp_imp**
- Export (imported, persistent)
  - Standard: **exp_imp_eav**

Considerations for Use of Persistent Views

- Higher performance
- Extra logic in action diagrams
- Fewer action diagrams
- Greater complexity for view matching
- Greater complexity for locking requirement
- Process Action Diagrams may look more “technical”
- Synthesis

Import/Export

- Values in the import view of the called action block will be returned to the matched view in the calling action block
- Can change the values of an import view
  - Transient View: through a MOVE, SET statement, or view matching
  - Persistent View: through a CREATE, READ, or UPDATE statement
- Performance gains through lower movement of data

View Properties

View Type	Persist	Imp & Exp	Locked
P-Step Import	No	Imp Only	N/A
P-Step Export	No	Exp Only	N/A
Act Blk Import	Yes/***No***	***Imp Only***/Exp	Lock/***Unlock***
Act Blk Export	Yes/***No***	***Exp Only***/Imp	Lock/***Unlock***
Local	No	N/A	N/A
Entity Action	N/A	N/A	N/A

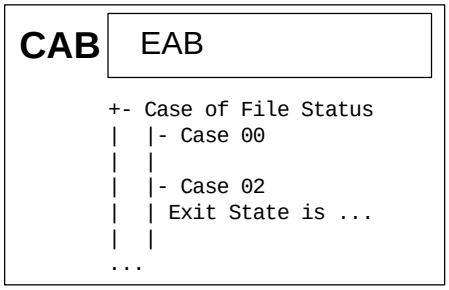
Note: default values are shown in italic

Situation

How can developers be insulated from some of the repetitive logic when using external action blocks?

Wrapping External Action Blocks

```
+-
|  ...
|  Use CAB
|  +- Case of Exit State
|  |- Case Open Error
|  | ...
```



- Calls the Common Action Block instead of External
- CAB serves as a pre- and post-translation layer
- Hides external action block considerations
- Saves on duplicate logic and views

Situation

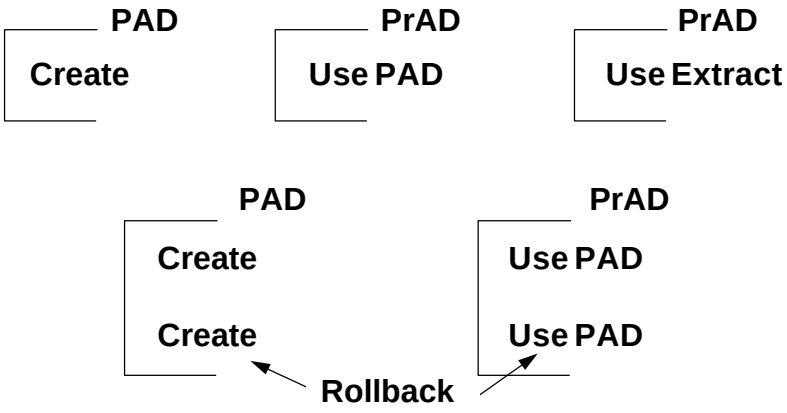
Have you ever had a new exit state being added to an action block give a hard to detect error? It shouldn't.

Proper Testing of Exit State

```
+-
| Exit State is Processing Okay
|  ...
|  Use Process Action Diagram
|  +- Case of Exit State
|  |- Case Processing Okay
|  |- Case Customer AE
|  | ...
|  |- Case Invalid Customer Type
|  | ...
|  |- Otherwise
|  | Exit State is Unexpected Exit State
|  +-
| ...
+-
```

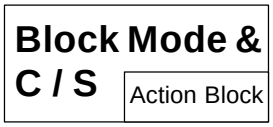

Proper Setting of Termination

- Work inside out, with each diagram correct

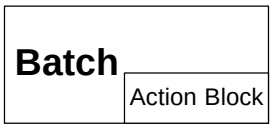


Situation

Action Block designs that are not compatible with the transaction style



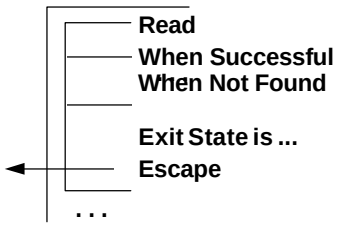
Single Error



Multiple Error

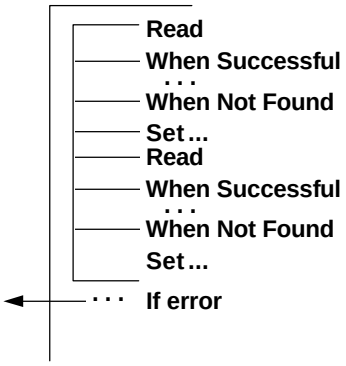
Errors and Action Block Design

- Traditional (Single)
  - On error Escape
  - Single error communicated by Exit State



Errors and Action Block Design

- Multiple
  - On error don't escape,
  - move CUD to the end,
  - group view of errors
    - »or
  - Duplicate logic
    - »or
  - Validation Action Block
- Error dependency
  - Like a compiler, one change affects a lot of errors



Road to Success...

Techniques for Increased Action Diagram Logic Reusability

Session 380

Greg Moll
Texas Instruments