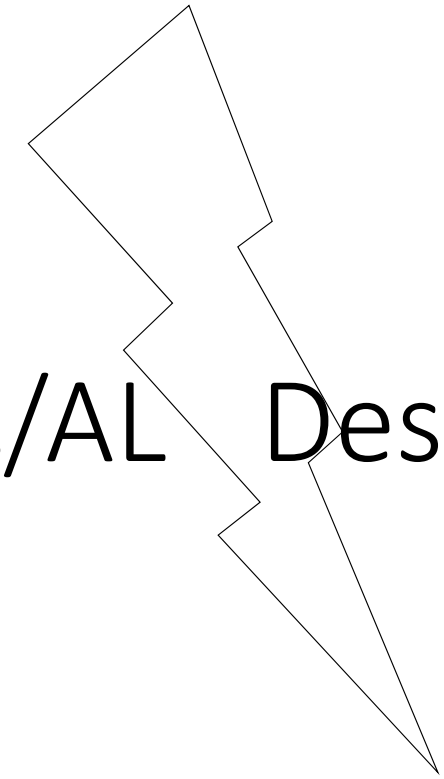
NAV 2015: THE LATEST APPLICATION CODE FROM A DESIGN PATTERNS PERSPECTIVE

Mostafa Balat, Bogdana Botez, Anders Larsen
(Microsoft MDCC)



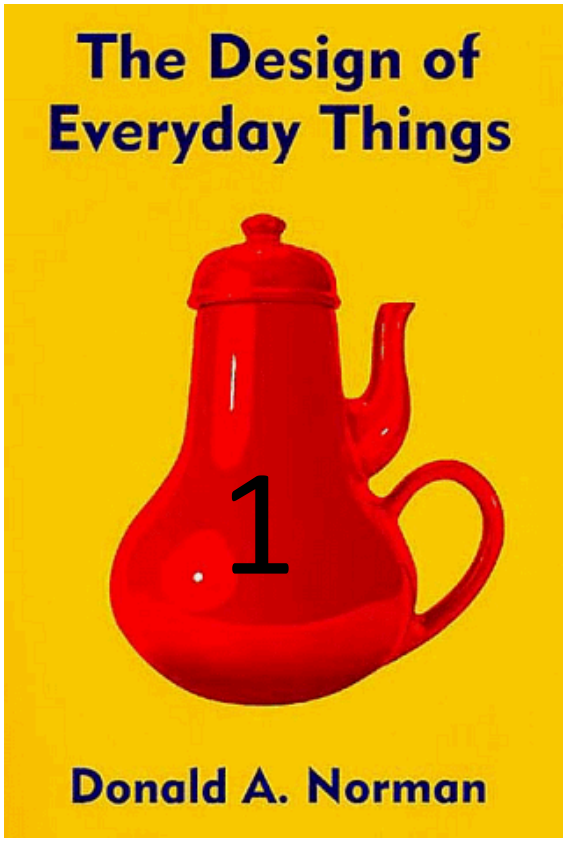
C/AL Design Patterns

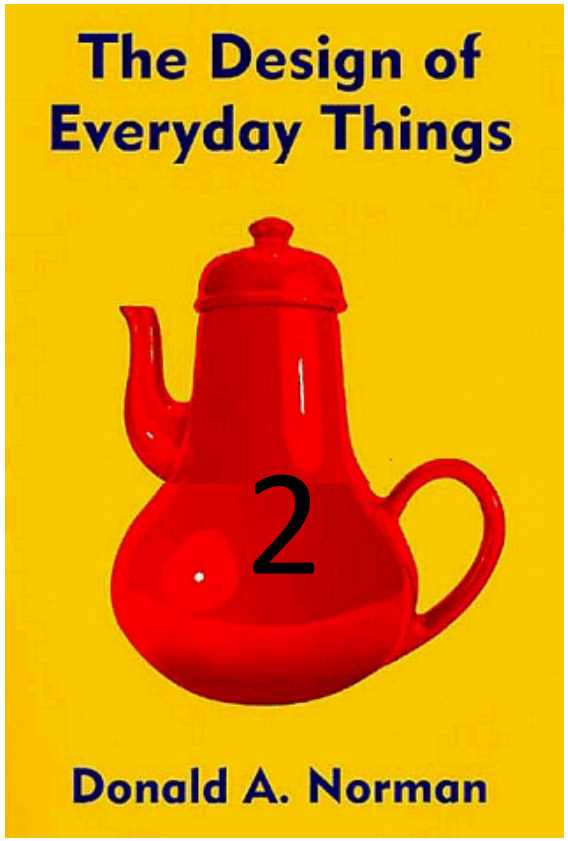
C/AL Developer

As a C/AL developer, this is your day

New Features	If only... … creating new features was quick and easy … you would know the proven solutions
Upgrade	If only... … MS developers wouldn't ruin your solution with each new version … upgrade was easy and FAST
Code maintenance	If only... … legacy code would follow a clearly stated intention … maintenance costs were low

Code Design


The Design of Everyday Things
1
Donald A. Norman


The Design of Everyday Things
2
Donald A. Norman

Agenda

WHAT is a Design Pattern?

WHY NAV Patterns?

WHAT NAV Patterns are we documenting?

Make your product

- **Extensible**
- **Upgradable**
- **Configurable**
- **Adaptable**
- **User-friendly**

NAV cookbook:

- **Cached Web Service Calls**
- **Copy Document**
- **SELECT DISTINCT using Queries**
- **C/AL guidelines**

WHO is part of the project?

WHAT next?

WHAT is a Design Pattern?

WHAT is a Design Pattern?	**WHY do we need Design Patterns?**	**WHAT NAV Patterns are we documenting?**	**WHO is part of the project?**	**WHAT Next?**

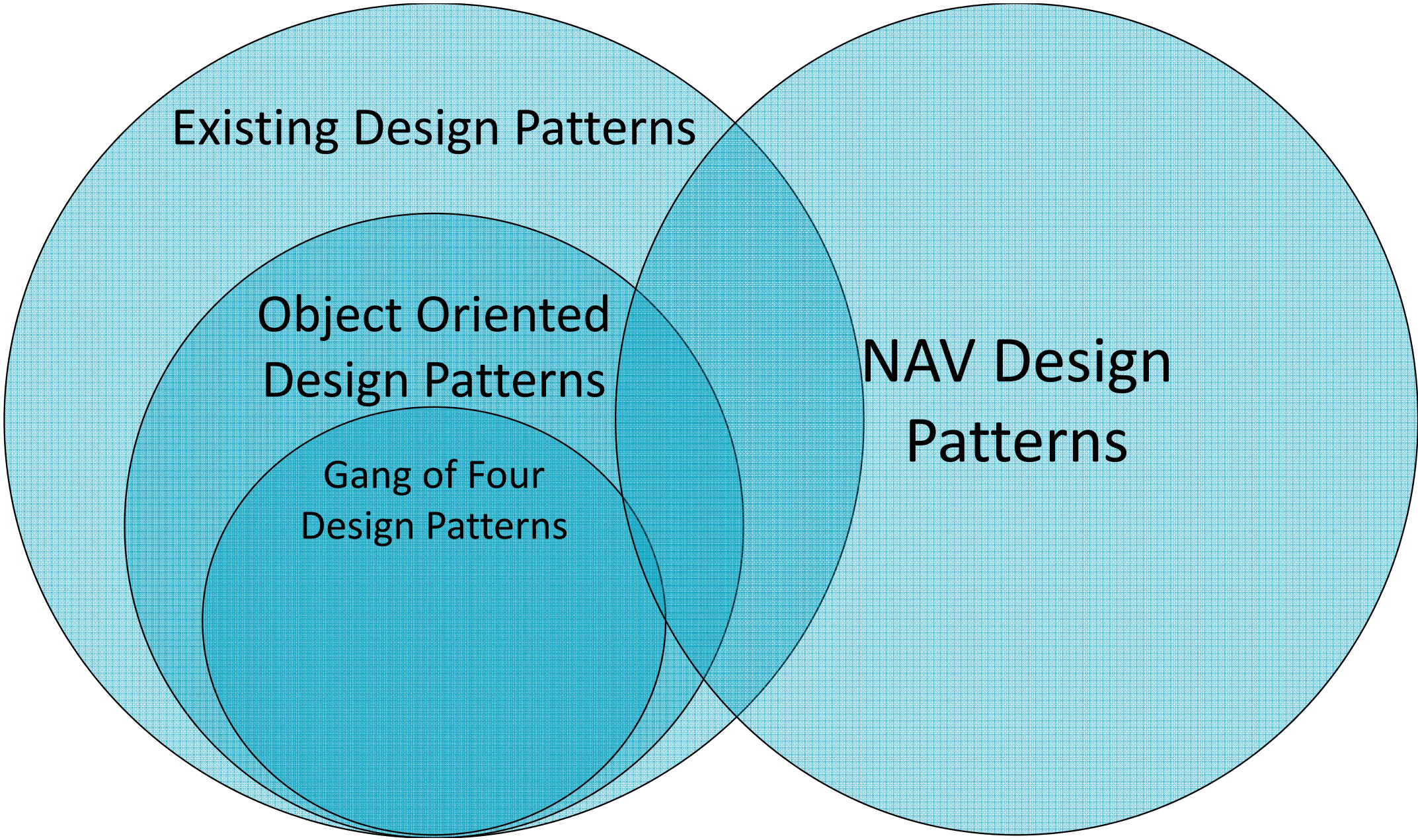
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

“Gang of Four”
Design Patterns

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

When the
“Gang of Four” was writing
Design Patterns, we knew that there were lots of software patterns other than object-oriented design patterns.

Ward Cunningham in the preface to "Analysis Patterns: Reusable Object Models" by Martin Fowler


Existing Design Patterns
Object Oriented Design Patterns
Gang of Four Design Patterns
NAV Design Patterns

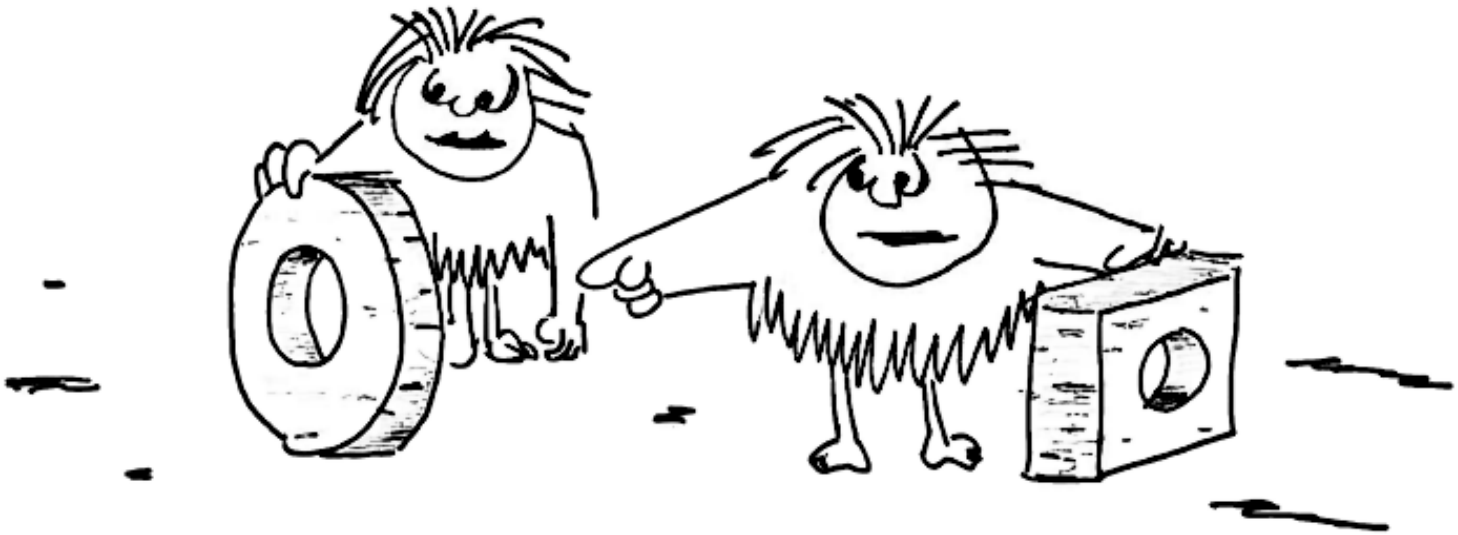
A pattern is an idea that has been useful in one practical context and will probably be useful in others.

Martin Fowler

WHY NAV Patterns?

WHAT is a Design Pattern?	**WHY NAV Patterns?**	WHAT NAV Patterns are we documenting?	WHO is part of the project?	WHAT next?

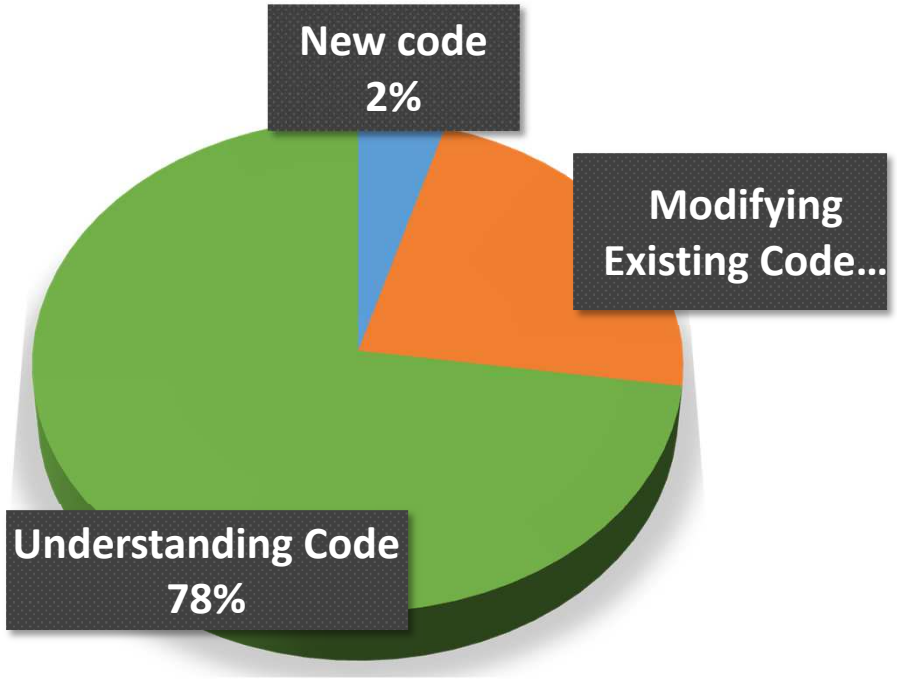
WHY NAV Patterns?



When understanding means rewriting

Time spent on:

New Code:	2%
Modifying Existing Code:	20%
Understanding Code:	78%



Avoid the MC Hammer code

Make your code extensible

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible	WHO is part of the project?	WHAT next?

Make Your Code Extensible

The Argument Table Pattern

Problem

- Using too many arguments in a function
- Hard to change signatures
- No function overloading in CAL
- Duplicating option definitions

Solution

- Group commonly used arguments as a table and use as a single argument.

Example 1

The Argument Table Pattern: Example 1

Bad example

Procedure definition

PROCEDURE FillInVATReturnData@1200001(VAR DeclarationID@1200000 : Code[20];VAR LineID@1200001 : Code[20];VAR PeerID@1200002 : Code[20]; VAR DocumentNo@1200003 : Code[20]; VAR NumberOfCopies@1200007 : Integer; VAR Uploaded@1200004 : Boolean; VAR Correction@1200005 : Boolean; VAR HasValidationErr@1200006 : Boolean);

Call

FillInVATReturnData(NoSeries, NextLineID, CustomerID, DocumentNo, SingleCopy, ???, ??, …., …)

The Argument Table Pattern: Example 1

Good example

New table

TAB 50003 VAT Return Data

Procedure definition

```
PROCEDURE FillInVATReturnData@1200001(VAR VATReturnData@1200000 : Record 50003);
```

Init default values

```
VATReturnData.INIT;
VATReturnData.NumberOfCopies := GetDefaultNumberOfCopies;
VATReturnData.Uploaded := FALSE;
```

Call

```
FillInVATReturnData(VATReturnData);
```

Example 2

The Argument Table Pattern: Example 2

Bad example

```
LOCAL PROCEDURE GetTableSyncSetupW1@3(
                                                OldTableId@1002 : Integer;
                                                VAR UpgradeTableId@1001 : Integer;
                                                VAR TableUpgradeMode@1000 : 'Check, Copy, Move, Force') : Boolean;
BEGIN
  CASE OldTableId OF
    DATABASE::"Sales Header":
      SetTableSyncSetup(0,TableUpgradeMode::Check,UpgradeTableId,TableUpgradeMode);
    DATABASE::"Posting Exch. Column Def":
      SetTableSyncSetup(104025,TableUpgradeMode::Copy,UpgradeTableId,TableUpgradeMode);
    DATABASE::"Payment Export Data":
      SetTableSyncSetup(0,TableUpgradeMode::Force,UpgradeTableId,TableUpgradeMode);
    ELSE
      EXIT(FALSE);
  END;

  EXIT(TRUE);
END;
```

The Argument Table Pattern: Example 2

Good example

```
PROCEDURE GetTableSyncSetupW1@3(VAR TableSynchSetup@1000 : Record 2000000135);
BEGIN
SetTableSyncSetup(DATABASE::"Sales Header",0,TableSynchSetup.Mode::Check);
  SetTableSyncSetup(DATABASE::"Posting Exch. Column Def",104025,TableSynchSetup.Mode::Copy);
  SetTableSyncSetup(DATABASE::"Payment Export Data",0,TableSynchSetup.Mode::Force);
END;
```

Overview: The Argument Table Pattern

Good

- Fewer arguments
- Clear intention
- Allows assigning default values
- Allows validation logic for parameters
- Easy to extend:
  - Easy to add new fields, no upgrade problem
  - No function signature change needed. Can be used where needed. Few lines of modification
- The table should be temporary

Bad

- More objects
- There is a limit to table encapsulation (cannot encapsulate records of another tables)

Make your code upgradable

WHAT is a Design Pattern?	WHY NAV Patterns?	**WHAT NAV Patterns are we documenting?** **Make your product** - **Extensible** - **Upgradable**	WHO is part of the project?	WHAT next?

The Hooks Pattern

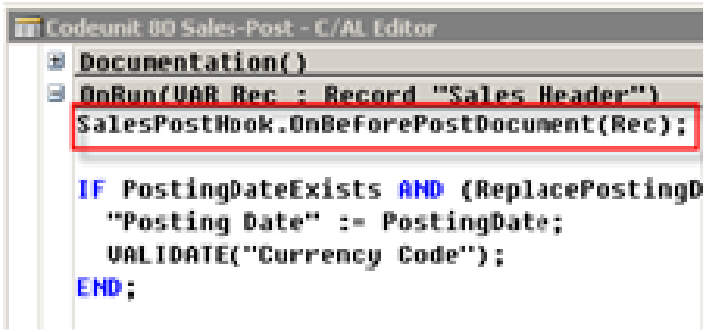
Problem: upgrade/merge conflicts
Solution: make your product upgradable, by minimizing your footprint

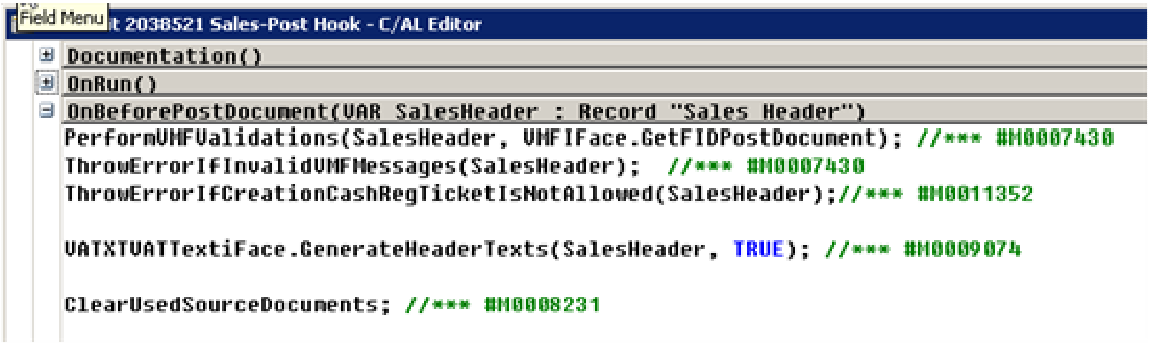
Hooks Pattern
by Eric Wauters

Two steps:

- name the places in the already existing code where customization is needed
- place all your business logic outside the already existing application code

The Hooks Pattern





https://community.dynamics.com/nav/w/designpatterns/117.hooks-pattern.aspx

The Hooks Pattern

Without the pattern, upgrade = 2 weeks

With the pattern, upgrade = 15 min

“Upgrading from 2013 to 2013R2, there was a change in codeunit 80 (piece of code moved to a function). There were lots of reactions from the community .. like <<Microsoft, please don't do that, now my code won't merge>>... I only had to move 1 line to that function...” (Eric Wauters)

Make your code configurable

WHAT is a Design Pattern?	WHY NAV Patterns?	**WHAT NAV Patterns are we documenting?** **Make your product** - **Extensible** - **Upgradable** - **Configurable**	WHO is part of the project?	WHAT next?

Make Your Product Configurable

Rules Table Pattern

Challenge

- Extending and maintaining hard-coded logic is expensive
- Minimize changes to existing code when adding new behavior
- Deploy changes to multiple tenants, plus customization for their businesses

Approach

- Data-driven changes are less expensive, easier to do

Rules Table

Example: Automatic Bank Reconciliation

Line No.	Statement Amount	Transaction Text	Payer Information	Additional Transaction Information
10000	-1500	Inv.10001	Cannon Group	
20000	-1750	Thank You!	Kennel	

Entry No.	Type	Amount	Document No.	Customer Name
1	Invoice	1500	10001	Cannon Group Plc.
2	Invoice	1500	99876	Cardoxy
3	Invoice	750	10201	Kennel
4	Invoice	2500	10210	Spotsmeyer's Furnishings
5	Invoice	2500	10211	Spotsmeyer's Furnishings
6	Invoice	5000	10212	Spotsmeyer's Furnishings

Rules Table

Bad example: Forest of IF Statements in Multiple Pages!

```
REPEAT
  CASE PostDesRec."Information Type" OF
    PostDesRec."Information Type"::"Description and Sundries":
      IF NOT SplitAccountnumber(PostDesRec.Description) THEN
        SplitInvoicenumber(PostDesRec.Description);
    PostDesRec."Information Type"::"Account No. Balancing Account":
      SplitAccountnumber(PostDesRec.Description);
    PostDesRec."Information Type"::"Name Acct. Holder":
      Name := PostDesRec.Description;
    PostDesRec."Information Type"::"Address Acct. Holder":
      Address := PostDesRec.Description;
    PostDesRec."Information Type"::"City Acct. Holder":
      City := PostDesRec.Description;
    PostDesRec."Information Type"::"Payment Identification":
      Identification := PostDesRec.Description;
  END;
UNTIL PostDesRec.NEXT = 0;

IF CBGStatementlineRec."Account No." = '' THEN BEGIN
  FOR i := 1 TO 5 DO BEGIN
    IF BankaccountNo[i] <> '' THEN BEGIN
      IF CBGStatementlineRec.Credit > 0 THEN BEGIN
        IF FindAccountnumber(BankaccountNo[i],TempRec."Source Type"::Customer,CBGStatementlineRec."Account No.") THEN BEGIN
          CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Customer;
          CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
          CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
          NumberOfLinesChanged := NumberOfLinesChanged + 1;
          RecChanged := TRUE;
        END;
      END ELSE BEGIN
        IF FindAccountnumber(BankaccountNo[i],TempRec."Source Type"::Vendor,CBGStatementlineRec."Account No.") THEN BEGIN
          CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Vendor;
          CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
          CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
          NumberOfLinesChanged := NumberOfLinesChanged + 1;
          RecChanged := TRUE;
        END;
      END;
    END;
  END;
  IF NOT RecChanged THEN BEGIN
    IF Name <> '' THEN BEGIN
      IF CBGStatementlineRec.Credit > 0 THEN BEGIN
        IF FindNAC(Name,Address,City,TempRec."Source Type"::Customer,CBGStatementlineRec."Account No.") THEN BEGIN
          CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Customer;
          CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
          CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
          NumberOfLinesChanged := NumberOfLinesChanged + 1;
          RecChanged := TRUE;
        END;
      END ELSE BEGIN
        IF FindNAC(Name,Address,City,TempRec."Source Type"::Vendor,CBGStatementlineRec."Account No.") THEN BEGIN
          CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Vendor;
          CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
```

Rules Table

Bad example 2: Argument table plus EXITs to avoid nesting

```
LOCAL GetMatchScore(VAR BankPmtApplRule : Record "Bank Pmt. Appl. Rule")
BankPmtApplRule.Score := 4000;
BankPmtApplRule."Match Confidence" := BankPmtApplRule."Match Confidence"::High;
IF (BankPmtApplRule."Related Party Matched" = BankPmtApplRule."Related Party Matched"::Fully) AND
  (BankPmtApplRule."Doc. No./Ext. Doc. No. Matched" = BankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::Yes) AND
  (BankPmtApplRule."Amount Incl. Tolerance Matched" = BankPmtApplRule."Amount Incl. Tolerance Matched"::"One Match") THEN
    EXIT;

BankPmtApplRule.Score -= 1;

IF (BankPmtApplRule."Related Party Matched" = BankPmtApplRule."Related Party Matched"::Fully) AND
  (BankPmtApplRule."Doc. No./Ext. Doc. No. Matched" = BankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::"Yes - Multiple") AND
  (BankPmtApplRule."Amount Incl. Tolerance Matched" = BankPmtApplRule."Amount Incl. Tolerance Matched"::"Multiple Match") THEN
    EXIT;

BankPmtApplRule.Score -= 1;

IF (BankPmtApplRule."Related Party Matched" = BankPmtApplRule."Related Party Matched"::Fully) AND
  (BankPmtApplRule."Doc. No./Ext. Doc. No. Matched" = BankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::"Yes - Multiple") AND
  (BankPmtApplRule."Amount Incl. Tolerance Matched" = BankPmtApplRule."Amount Incl. Tolerance Matched"::"Multiple Match") THEN
    EXIT;
```

In total, there are 3 full screens of code like this!

Rules Table - Overview

Good example: Implement the matching rules as a table

Match Confidence	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	Fully	No	One Match
Match Confidence	**Related Party Matched**	**Document No. / Ext. Document No. Matched**	**Number of Entries Within Amount Tolerance Found**
High	Fully	No	One Match
Medium	Partially	Yes - Multiple	Not Considered
Low	Fully	Yes	No Matches
Low	Partially	Yes	No Matches

Transactions from Bank Statement File

Statement Amount	Transaction Text	Payer Information	Additional Transaction Information
-1500	Invoice on the date	Cannon Group	

Open Ledger Entries

Entry No.	Type	Amount	Document No.	Customer Name
1	Invoice	1500	10001	Cannon Group Plc.

Rules Table – Implementation

```
LOCAL FindMatchingEntry(VAR BankPmtApplRule : TEMP
// Set Parameters - use Argument pattern
RelatedPartyMatching(BankPmtApplRule,...);
DocumentMatching(BankPmtApplRule,...);
AmountInclToleranceMatching(BankPmtApplRule,...);

// Get/find the score
Score := TempBankPmtApplRule.GetBestMatchScore(BankPmtApplRule);
```

Match Confidence	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	Fully	No	One Match

```
GetBestMatchScore(ParameterBankPmtApplRule : Record "Bank Pmt. Appl. Rule") : Integer
CLEAR(Rec);
SETCURRENTKEY(Score);
ASCENDING(FALSE);

SETFILTER("Related Party Matched",'%1|%2',
  ParameterBankPmtApplRule."Related Party Matched",
  ParameterBankPmtApplRule."Related Party Matched"::"Not Considered");

SETFILTER("Doc. No./Ext. Doc. No. Matched",'%1|%2',
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched",
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::"Not Considered");

SETFILTER("Amount Incl. Tolerance Matched",'%1|%2',
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched",
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched"::"Not Considered");

IF FINDFIRST THEN
  EXIT(Score);

EXIT(0);
```

Rules Table – Adding More Rules

Modify or add rows to the table. No code modifications needed.

Match Confidence	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found
High	Fully	Yes - Multiple	One Match
High	Fully	Yes - Multiple	Multiple Matches
High	Fully	No	One Match
Medium	Fully	Yes - Multiple	Not Considered
Medium	Fully	Yes	Not Considered
Medium	Fully	No	Multiple Matches
Medium	Partially	Yes - Multiple	Not Considered
Low	Fully	Yes	No Matches
Low	Partially	Yes	No Matches
Low	Fully	No	No Matches
Low	Partially	No	One Match

Rules Table – Extending Criteria

1. Adding additional criterion: Add a column to the table.

Match Confidence	Related Party Matched	Document No. / Ext. Document No. Matched	Number of Entries Within Amount Tolerance Found	Due Date
High	Fully	Yes - Multiple	One Match	No
High	Fully	Yes - Multiple	Multiple Matches	No
High	Fully	No	One Match	No
Medium	Fully	Yes - Multiple	Not Considered	No
Medium	Fully	Yes	Not Considered	No
Medium	Fully	No	Multiple Matches	No
Medium	Partially	Yes - Multiple	Not Considered	Yes
Low	Fully	Yes	No Matches	No
Low	Partially	Yes	No Matches	No
Low	Fully	No	No Matches	Yes
Low	Partially	No	One Match	Yes

Rules Table – Extending Criteria

2. Adding additional criterion: Update code in two places

```
LOCAL FindMatchingEntry(VAR BankPmtApplRule : TEMPORARY Record "Bank Pmt. Appl. Rule")
// Set Parameters - use Argument pattern
RelatedPartyMatching(BankPmtApplRule,...);
DocumentMatching(BankPmtApplRule,...);
AmountInclToleranceMatching(BankPmtApplRule,...);

// Add the Document overdue logic - Can use the Hook pattern
DocumentIsOverdueMatching(BankPmtApplRule,...);

// Get/find the score
Score := TempBankPmtApplRule.GetBestMatchScore(BankPmtApplRule);
```

```
GetBestMatchScore(ParameterBankPmtApplRule : Record "Bank Pmt. Appl. Rule") : Integer
CLEAR(Rec);
SETCURRENTKEY(Score);
ASCENDING(FALSE);

SETFILTER("Related Party Matched",'%1|%2',
  ParameterBankPmtApplRule."Related Party Matched",
  ParameterBankPmtApplRule."Related Party Matched"::"Not Considered");

SETFILTER("Doc. No./Ext. Doc. No. Matched",'%1|%2',
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched",
  ParameterBankPmtApplRule."Doc. No./Ext. Doc. No. Matched"::"Not Considered");

SETFILTER("Amount Incl. Tolerance Matched",'%1|%2',
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched",
  ParameterBankPmtApplRule."Amount Incl. Tolerance Matched"::"Not Considered");

// Add new filter for OverDue
```

Rules Table

Other Implementations

- Data Exchange Framework
- Rapidstart Configuration Templates
- Rapidstart – Field Mappings and Text Processing Rules, etc.

Rules Table

Good

- Extensible
- Readable
- Easier to maintain
- Simpler to upgrade
- Faster to deploy

Bad

- Tables can grow quickly making it difficult to understand

Make your code adaptable to user needs

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable	WHO is part of the project?	WHAT next?

Make Your Product Adaptable

Presentation Model Pattern

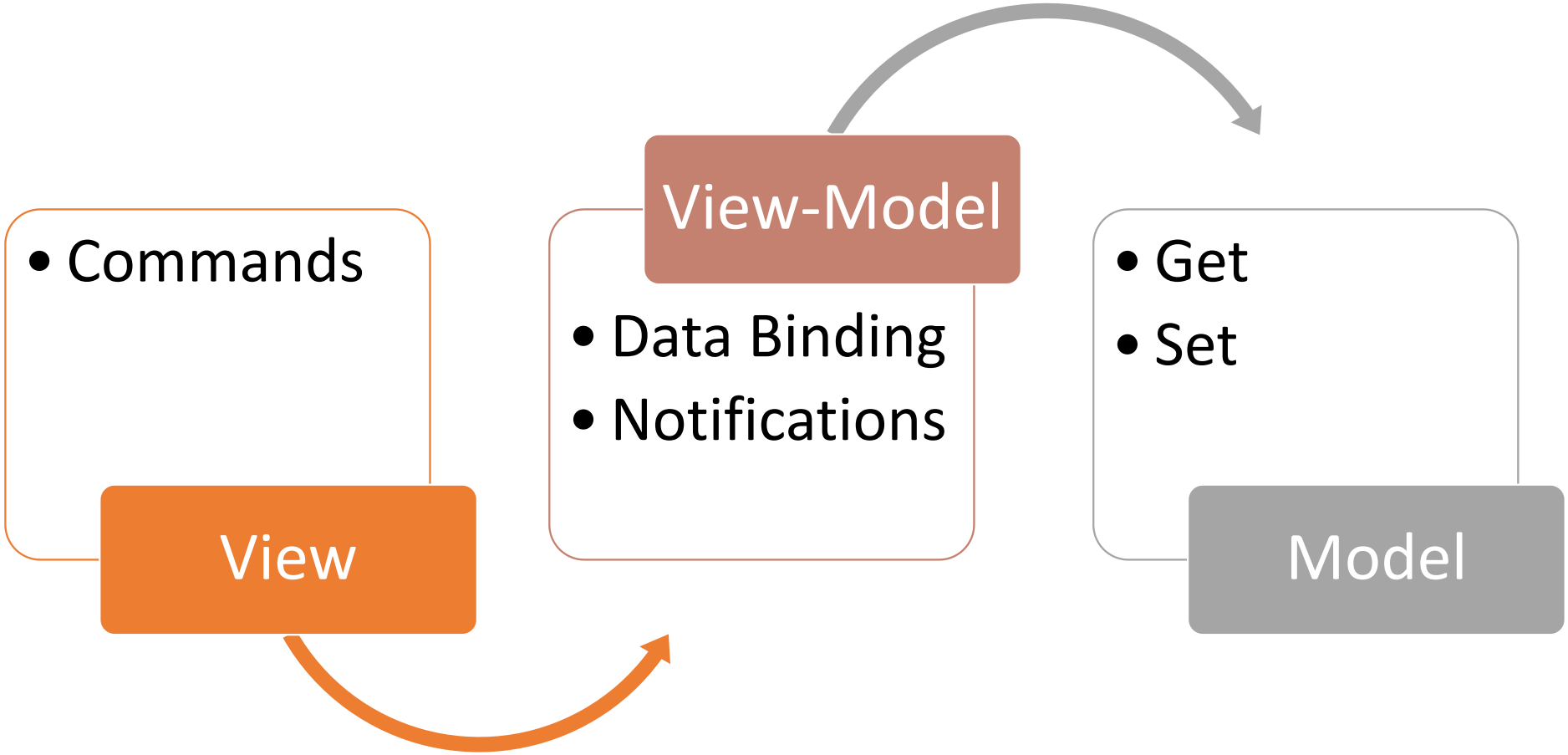
Challenge

- Using same table for real and temporary may undesirably impact other related data
- Validation triggers are partially or not needed for temporary processing
- UI should be business logic-free

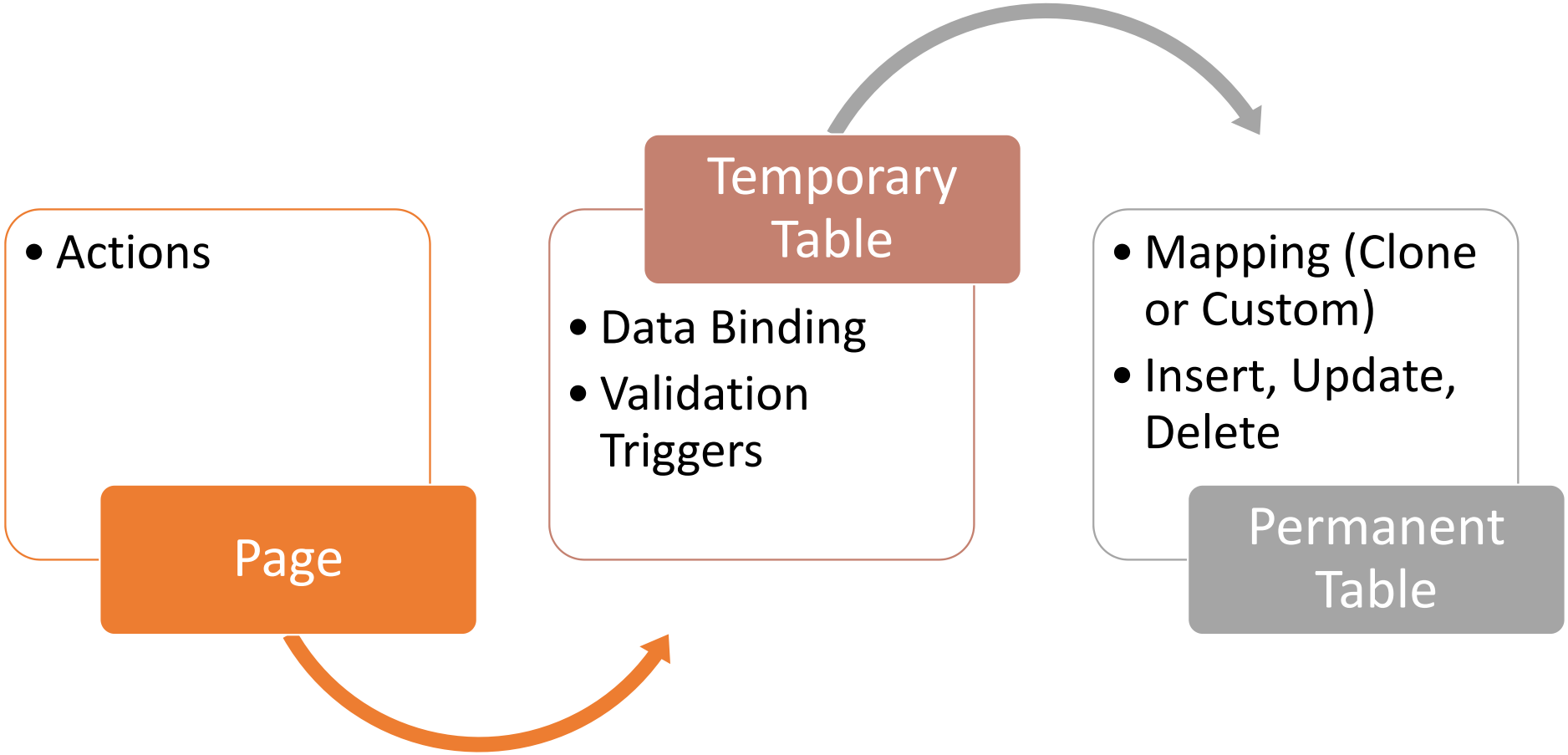
Approach

- Create a new table with replica or custom mapping for temporary data

Presentation Model Pattern



Presentation Model – Implementation



Presentation Model in NAV 2015

Bank Reconciliation

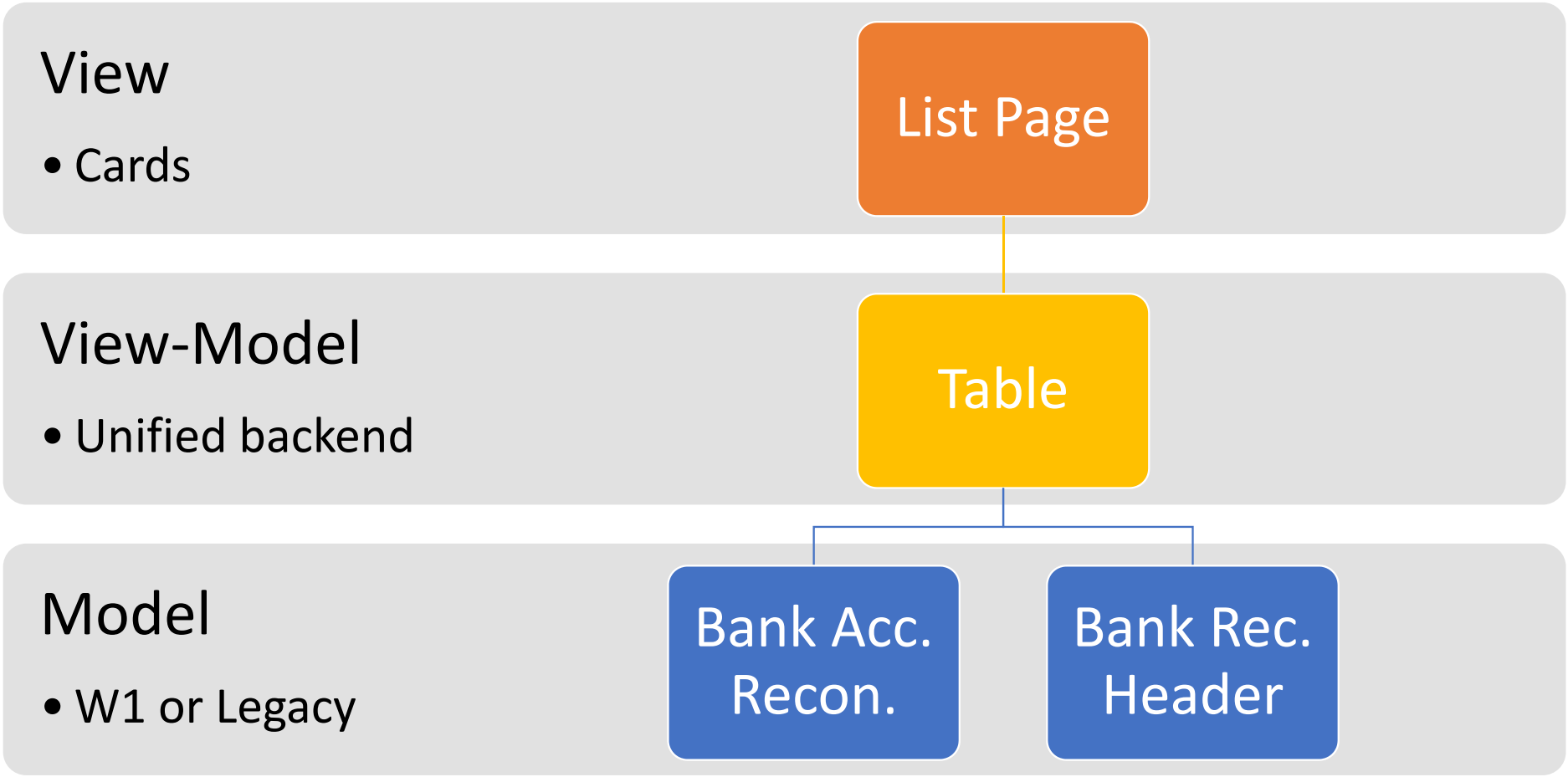
Challenge

- W1 implementation with improvements overshadows localization
- Import bank statement file, auto-match statement lines to Ledger Entries
- Customers decide when to migrate their business process off legacy localization

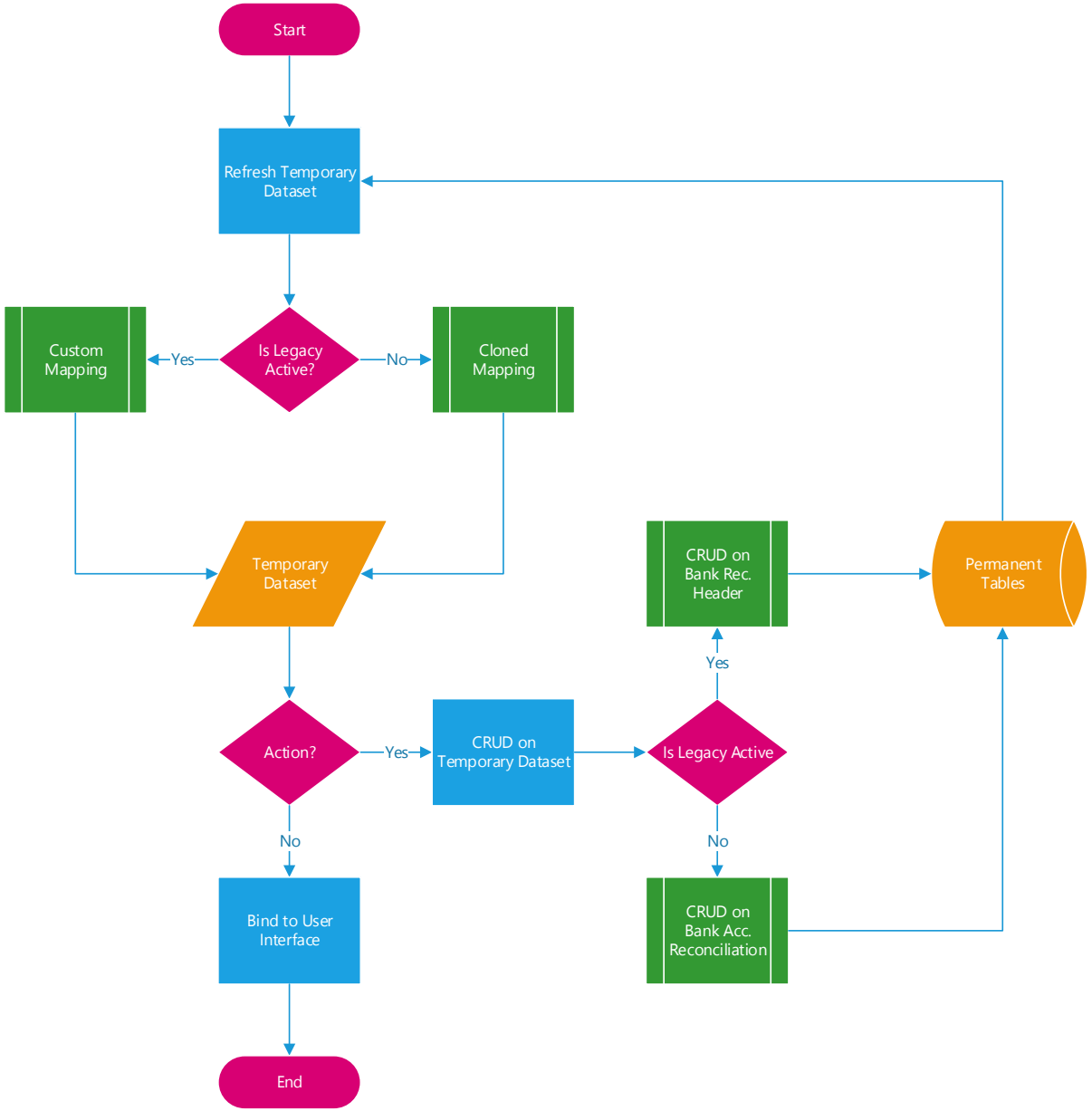
Approach

- Display both old and new data through the same starting point
- Statistical cues reflect old or new data, based on user's choice
- Provide a switch to move from legacy to universal implementation

Presentation Model – Bank Reconciliation



Presentation Model – Example



Presentation Model – Bank Reconciliation

1. Get temporary copies of the Bank Acc. Reconciliation table

```
GetTempCopy(VAR BankAccReconciliation : Record "Bank Acc. Reconciliation")
IF BankAccReconciliation.HASFILTER THEN
  COPYFILTERS(BankAccReconciliation);

SETRANGE("Statement Type","Statement Type"::"Bank Reconciliation");
IF NOT FINDSET THEN
  EXIT;

REPEAT
  BankAccReconciliation := Rec;
  BankAccReconciliation.INSERT;
UNTIL NEXT = 0;
```

Presentation Model – Bank Reconciliation

2. Get temporary copies of the Bank Rec. Header table

```
GetTempCopyFromBankRecHeader(VAR BankAccReconciliation : Record "Bank Acc. Reconciliation")
IF BankAccReconciliation.HASFILTER THEN BEGIN
  BankRecHeader.SETFILTER("Bank Account No.",BankAccReconciliation.GETFILTER("Bank Account No."));
  BankRecHeader.SETFILTER("Statement No.",BankAccReconciliation.GETFILTER("Statement No."));
  BankRecHeader.SETFILTER("Statement Date",BankAccReconciliation.GETFILTER("Statement Date"));
  BankRecHeader.SETFILTER("Statement Balance",BankAccReconciliation.GETFILTER("Balance Last Statement"));
END;

IF NOT BankRecHeader.FINDSET THEN
  EXIT;

REPEAT
  BankAccReconciliation."Statement Type" := BankAccReconciliation."Statement Type"::"Bank Reconciliation";
  BankAccReconciliation."Bank Account No." := BankRecHeader."Bank Account No.";
  BankAccReconciliation."Statement No." := BankRecHeader."Statement No.";
  BankAccReconciliation."Statement Date" := BankRecHeader."Statement Date";
  BankAccReconciliation."Balance Last Statement" := BankRecHeader."Statement Balance";
  BankAccReconciliation."Statement Ending Balance" := BankRecHeader.CalculateEndingBalance;
  BankAccReconciliation.INSERT;
UNTIL BankRecHeader.NEXT = 0;
```

Presentation Model – Bank Reconciliation

3. Bind the temporary copies to the right set of Card page

```
NewRec - OnAction()
BankReconciliationMgt.New(Rec,UseSharedTable);

EditRec - OnAction()
BankReconciliationMgt.Edit(Rec,UseSharedTable);

RefreshList - OnAction()
Refresh;

DeleteRec - OnAction()
IF NOT CONFIRM(DeleteConfirmQst) THEN
  EXIT;

BankReconciliationMgt.Delete(Rec);
Refresh;

Post - OnAction()
BankReconciliationMgt.Post(Rec,CODEUNIT::"Bank Acc. Recon. Post (Yes/No)",CODEUNIT::"Bank Rec.-Post (Yes/No)");
Refresh;

PostAndPrint - OnAction()
BankReconciliationMgt.Post(Rec,CODEUNIT::"Bank Acc. Recon. Post+Print",CODEUNIT::"Bank Rec.-Post + Print");
Refresh;

LOCAL Refresh()
DELETEALL;
BankReconciliationMgt.Refresh(Rec);
```

Presentation Model – Bank Reconciliation

3. Bind the temporary copies to the right set of Card page

```
Edit(VAR BankAccReconciliation : Record "Bank Acc. Reconciliation";ShareTable : Boolean)
StatementType := BankAccReconciliation."Statement Type";
BankAccountNo := BankAccReconciliation."Bank Account No.";
StatementNo := BankAccReconciliation."Statement No.";

IF AutoMatchSelected AND BankAccReconciliation2.GET(StatementType,BankAccountNo,StatementNo) THEN BEGIN
  BankAccReconciliationCard.SetSharedTempTable(BankAccReconciliation);
  IF ShareTable THEN
    BankAccReconciliationCard.SETRECORD(BankAccReconciliation2);
  BankAccReconciliationCard.RUN;
END;

IF (NOT AutoMatchSelected) AND BankRecHeader.GET(BankAccountNo,StatementNo) THEN BEGIN
  IF ShareTable THEN
    BankRecWorksheet.SetSharedTempTable(BankAccReconciliation);
  BankRecWorksheet.SETRECORD(BankRecHeader);
  BankRecWorksheet.RUN;
END;
```

Presentation Model Design Pattern

Good

- Adjust the representation of the same data to different needs
- Enable basic and advanced scenarios (regular Customer card vs. Mini Customer card)
- Migrate users easily to new implementations with less effort

Bad

- Mainly for the user interface. Otherwise, Queries should be used.

Make your code user friendly

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable - User Friendly	WHO is part of the project?	WHAT next?

Make your code user friendly

.NET Exception Handling

Challenge

- Need to reuse .NET classes in C/AL
- Unhandled exceptions annoy users
- No Try-Catch clause in C/AL

Approach

- .NET exception handling in C/AL

.NET Exception Handling

Example

```
IF NOT CheckFileExistence.RUN(TempBlob) THEN BEGIN
 ExceptionHandler.Collect;

  CASE TRUE OF      // CATCH STATEMENTS
    ExceptionHandler.TryCastToType(GETDOTNETTYPE(FileNotFoundException)):
      CheckFileExistence.ImportFile(TempBlob);
    ExceptionHandler.TryCastToType(GETDOTNETTYPE(UnauthorizedAccessException)):
      CheckFileExistence.UploadFileFromClient(TempBlob,FileName);
    ELSE      // CATCH Exception
      DotNetExceptionHandler.Rethrow;
  END;
END;
```

.NET Exception Handling in NAV 2015

Web Service Management

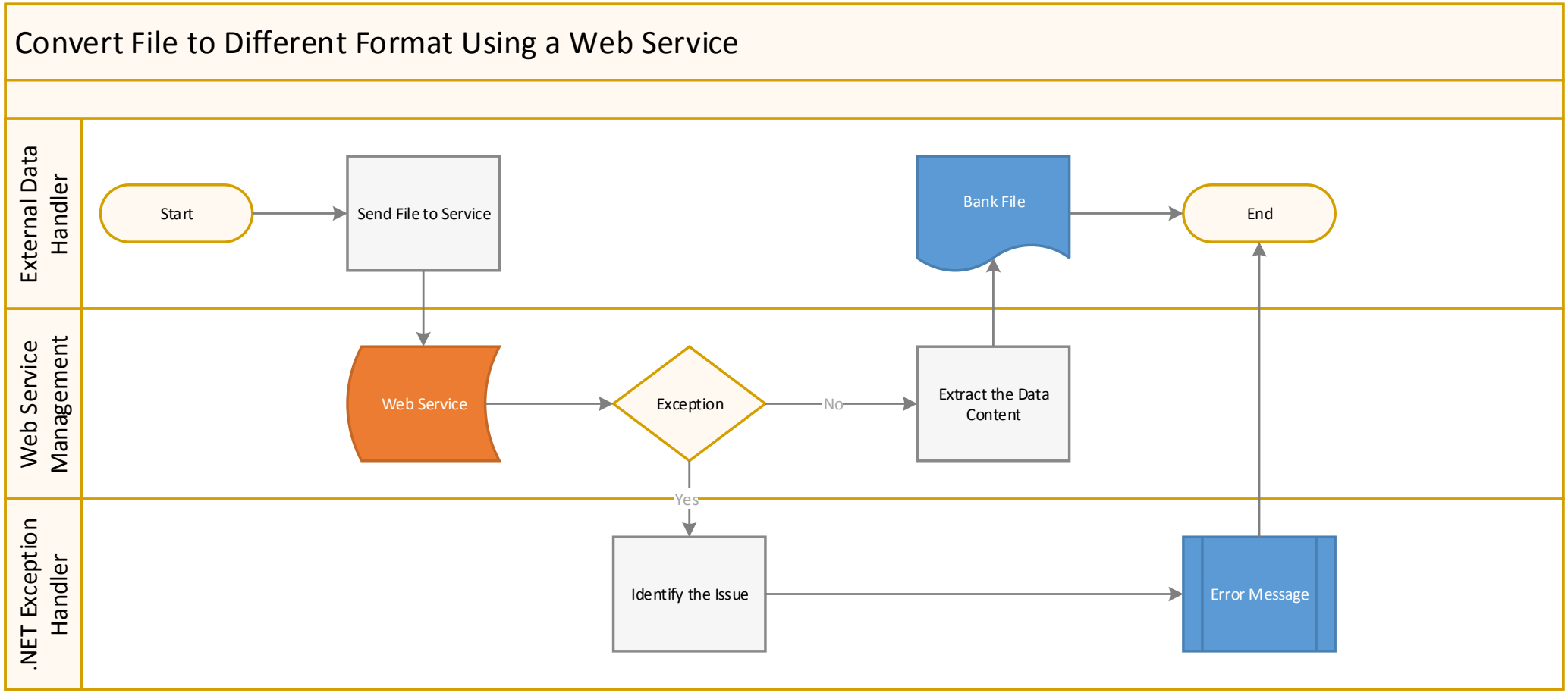
1. Construct a web request.
2. Connect to a web service.
3. Handle any web exceptions.
4. Extract the error response.
5. Display a user-friendly error.

Feature: Bank Data Conversion

- Payment Export
- Bank Statement Import
- Bank Name Lookup

Foundation for C/AL-powered Integration to Service-oriented Architecture Deployment

.NET Exception Handling – Example



Example – External Data Handler

```
LOCAL SendDataToConversionService(VAR PaymentFileTempBlob : Record TempBlob;VAR BodyTempBlob : Record TempBlob;Posti
PrepareSOAPRequestBody(BodyTempBlob);

BankDataConvServiceSetup.GET;
BodyTempBlob.Blob.CREATEINSTREAM(BodyInStream);

WebServiceRequestMgt.SetGlobals(BodyInStream,
  BankDataConvServiceSetup."Service URL",BankDataConvServiceSetup."User Name",BankDataConvServiceSetup.GetPassword);

IF NOT WebServiceRequestMgt.RUN THEN
  WebServiceRequestMgt.ProcessFaultResponse;

WebServiceRequestMgt.GetResponseContent(ResponseInStream);

CheckIfErrorsOccurred(ResponseInStream,PostingExchEntryNo);

ReadContentFromResponse(PaymentFileTempBlob,ResponseInStream);
```

Example – Web Service Management

```
ProcessFaultResponse()
DotNetExceptionHandler.Collect;

IF NOT DotNetExceptionHandler.CastToType(WebException,GETDOTNETTYPE(WebException)) THEN
  DotNetExceptionHandler.Rethrow;

IF NOT WebException.Status.Equals(WebExceptionStatus.ProtocolError) THEN
  ERROR(WebException.Message);

ResponseInputStream := WebException.Response.GetResponseStream;
DebugLogStreamToTempFile(ResponseInputStream, 'WebExceptionResponse',TempDebugLogTempBlob);

HttpWebResponseError := WebException.Response;
IF NOT (HttpWebResponseError.StatusCode.Equals(HttpStatusCode.Found) OR
        HttpWebResponseError.StatusCode.Equals(HttpStatusCode.InternalServerError))
THEN
  ERROR(WebException.Message);

XmlDoc := XmlDoc.XmlDocument;
XmlDoc.Load(ResponseInputStream);
ERROR(XMLDOMMgt.FindNodeTextWithNamespace(XmlDoc.DocumentElement,FaultStringXmlPathTxt,'soap',SoapNamespaceTxt));
```

Example – Exception Handler

```
Catch(VAR Exception : DotNet "System.Exception";Type : DotNet "System.Type")
Collect;
IF NOT CastToType(Exception,Type) THEN
  Rethrow;

Collect()
OuterException := GETLASTERROROBJECT;

LOCAL IsCollected() : Boolean
EXIT(NOT ISNULL(OuterException));

TryCastToType(Type : DotNet "System.Type") : Boolean
EXIT(CastToType(Exception,Type));

CastToType(VAR Exception : DotNet "System.Exception";Type : DotNet "System.Type") : Boolean
IF NOT IsCollected THEN
  EXIT(FALSE);

Exception := OuterException.GetBaseException;
IF ISNULL(Exception) THEN
  EXIT(FALSE);

IF Type.Equals(Exception.GetType) THEN
  EXIT(TRUE);

EXIT(FALSE);

Rethrow()
IF NOT IsCollected THEN
  EXIT;

IF ISNULL(OuterException.InnerException) THEN
  ERROR(OuterException.Message);

ERROR(OuterException.InnerException.Message);
```

.NET in C/AL vs. Add-Ins

Good

- Easier to customize (by in-house and partner developers alike)
- Easier to deploy (FOB or text)
- Easier to upgrade (no external dependencies)
- Reuse of knowledge (written in C/AL)

Bad

- OnRun usage trigger might be tricky in case of codeunit-nested invocation.

NAV cookbook

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable - User Friendly	WHO is part of the project?	WHAT next?
		NAV cookbook: - Cached Web Service Calls		

Reuse Data from External Services

Cached Web Service Calls

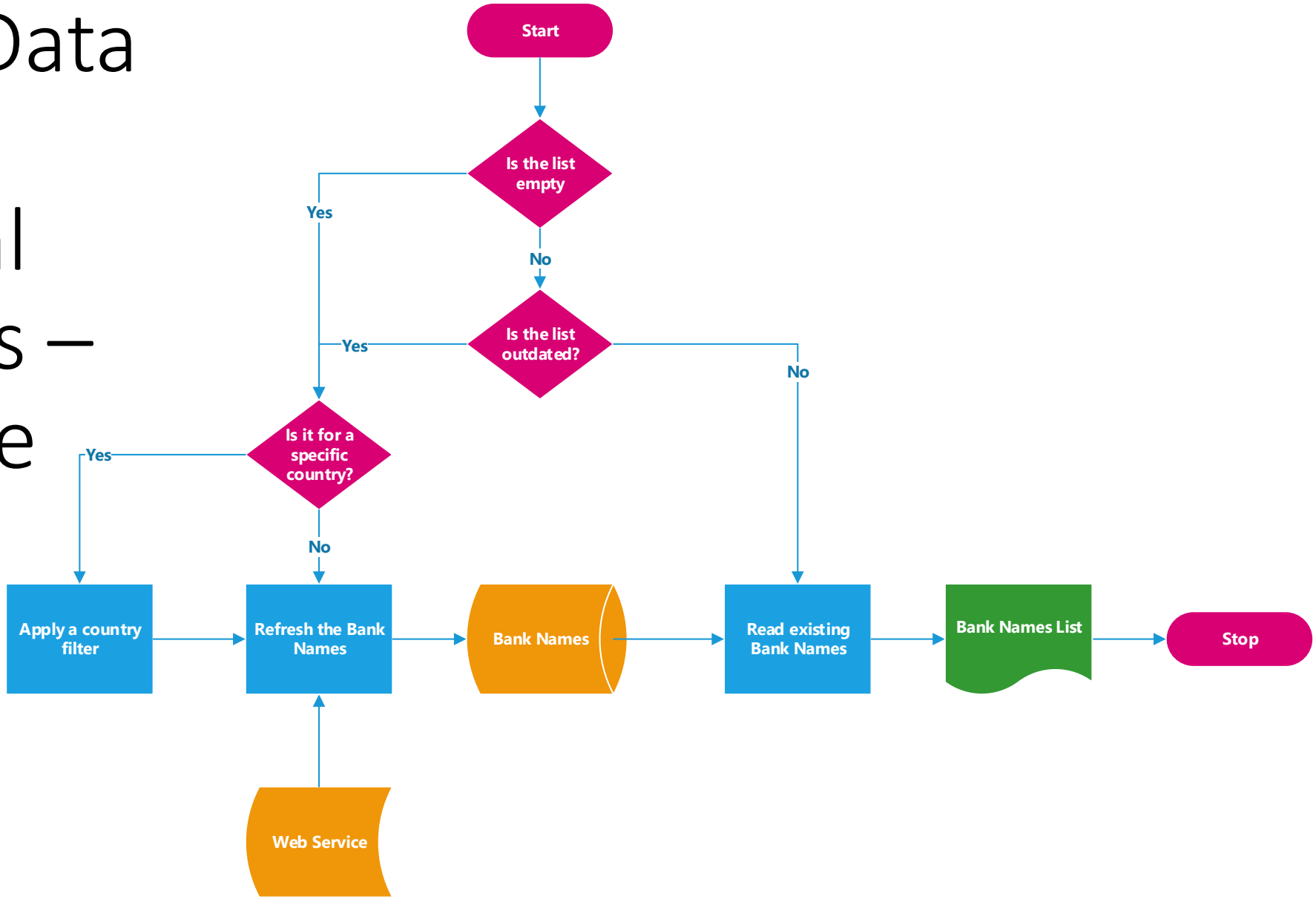
Challenge

- Reference data needs to be up-to-date.
- Connected model, network latency, service downtime
- Disconnected model, out of sync, maintainability

Approach

- Retrieve and cache the data periodically

Reuse Data from External Services – Example



Reuse Data from External Services in NAV

Bank Names Table

- Host the list of bank names cached from a web service
- Uses a timestamp to recognize the last refresh time

Bank Names List

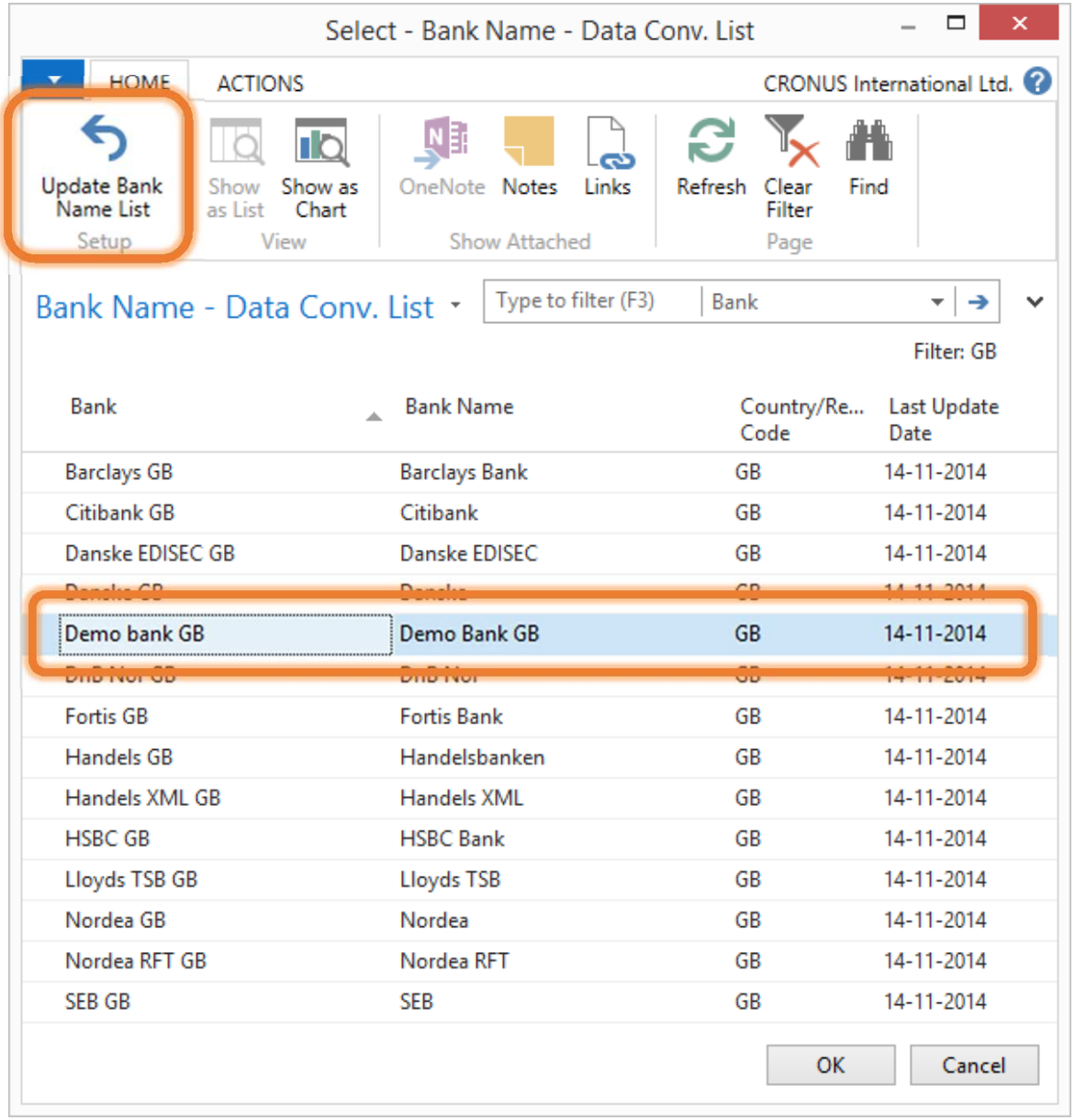
- Display the bank names in full or lookup modes
- Auto-refresh the data for one or more Country/Region Codes
- Provide an action to the user for forcing a data refresh

Feature: Bank Data Conversion

- Payment Export

Reuse Data from External Services in NAV

Bank Names List



Reuse Data from External Services – Example

OnOpenPage

```
BEGIN
  CountryRegionCode := IdentifyCountryRegionCode(Rec,GETFILTER("Country/Region Code"));

  IF BankDataConvBank.ISEMPTY THEN BEGIN
    ImpBankListExtDataHndl.GetBankListFromConversionService(HideErrors,CountryRegionCode,ShortTimeout);
    EXIT;
  END;

  RefreshBankNamesOlderThanToday(CountryRegionCode);
END;
```

Reuse Data from External Services – Example

RefreshBankNamesOlderThanToday

```
BEGIN
  IF CountryRegionCode <> '' THEN
    BankDataConvBank.SETFILTER("Country/Region Code",CountryRegionCode);
  BankDataConvBank.SETFILTER("Last Update Date",'<%1',TODAY);
  IF BankDataConvBank.FINDFIRST THEN
    ImpBankListExtDataHndl.GetBankListFromConversionService(ShowErrors,CountryRegionCode,Timeout);
END;
```

Reuse Data from External Services

Good

- Avoid connection timeout, service downtime
- Working across devices, users may have limited bandwidth
- Data is ‘relatively fresh’ according to the business expectations

Bad

- Dependency on an external service is as good as the service uptime is

Copy Document

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable - User Friendly	WHO is part of the project?	WHAT next?
		NAV cookbook: - Cached Web Service Calls - Copy Document		

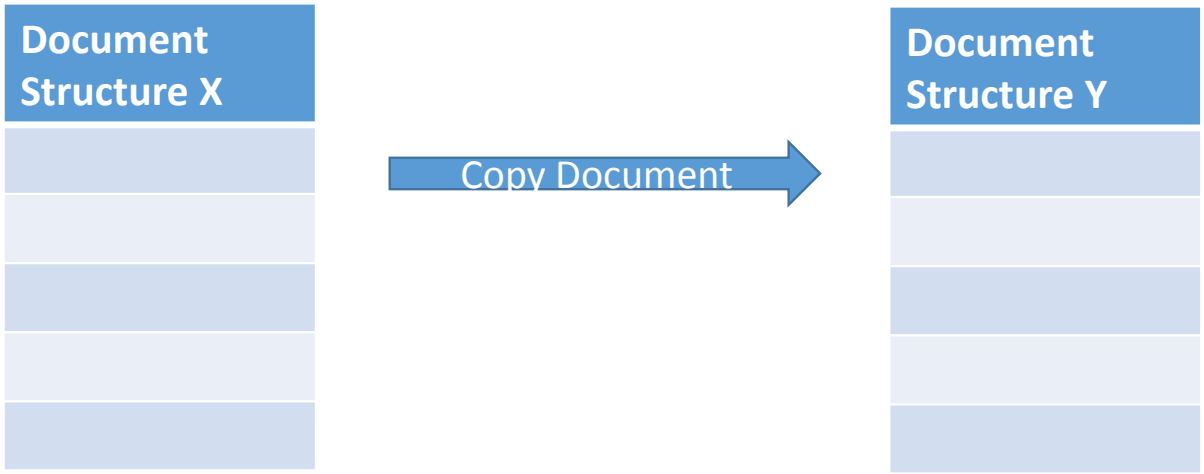
Copy Document

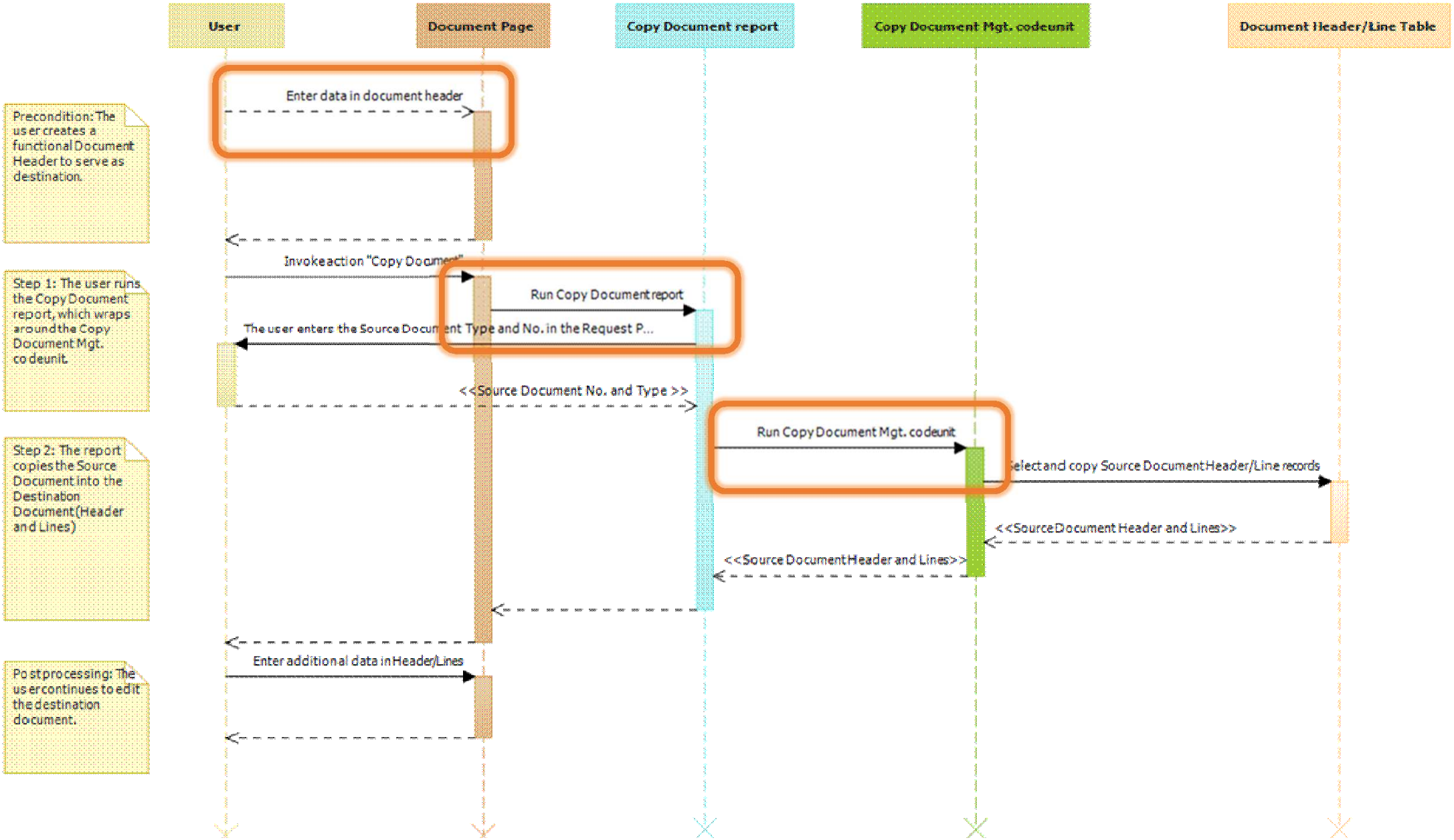
Challenge

- When a user wants to create a entity of a documents structure based on a existing entity of the same document structure.
- When a user want to be able to copy from one Document structure to another

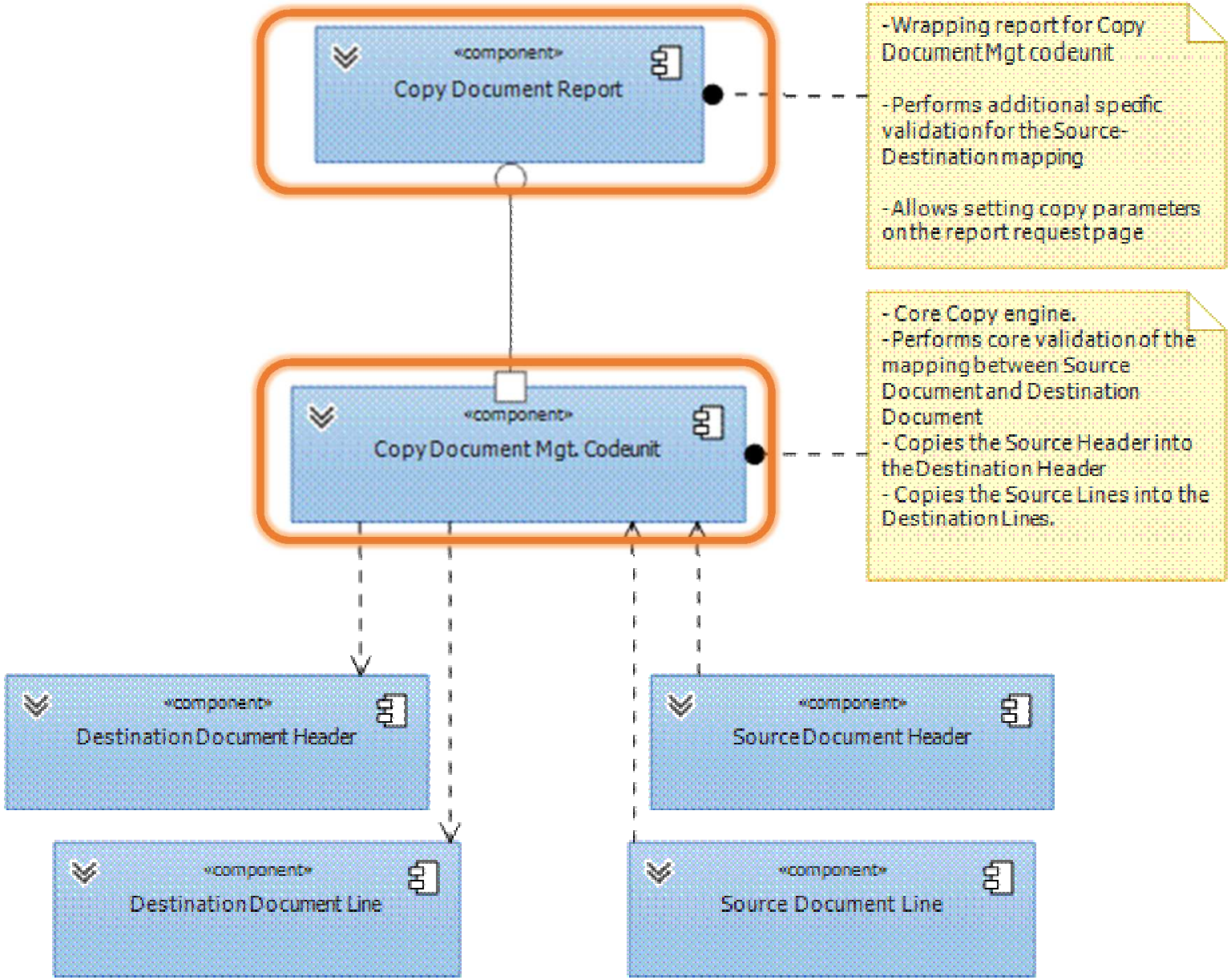
Approach

- Create a codeunit for the remapping of the fields




User
Document Page
Copy Document report
Copy Document Mgt. codeunit
Document Header/Line Table
Precondition: The user creates a functional Document Header to serve as destination.
Enter data in document header
Step 1: The user runs the Copy Document report, which wraps around the Copy Document Mgt. codeunit.
Invoke action "Copy Document"
Run Copy Document report
The user enters the Source Document Type and No. in the Request P...
<<Source Document No. and Type >>
Step 2: The report copies the Source Document into the Destination Document (Header and Lines)
Run Copy Document Mgt. codeunit
Select and copy Source Document Header/Line records
<<Source Document Header and Lines>>
<<Source Document Header and Lines>>
Post processing: The user continues to edit the destination document.
Enter additional data in Header/Lines

Copy Document



Copy Document

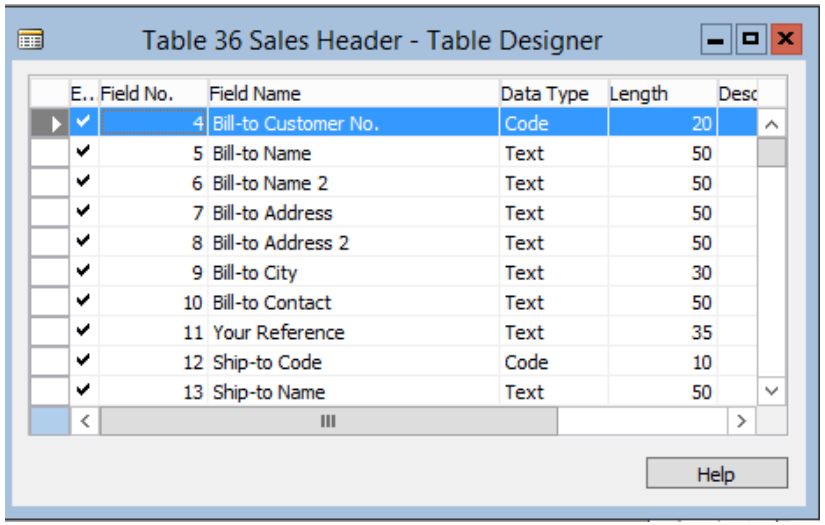
- Implicit implementation patterns
- Validation order

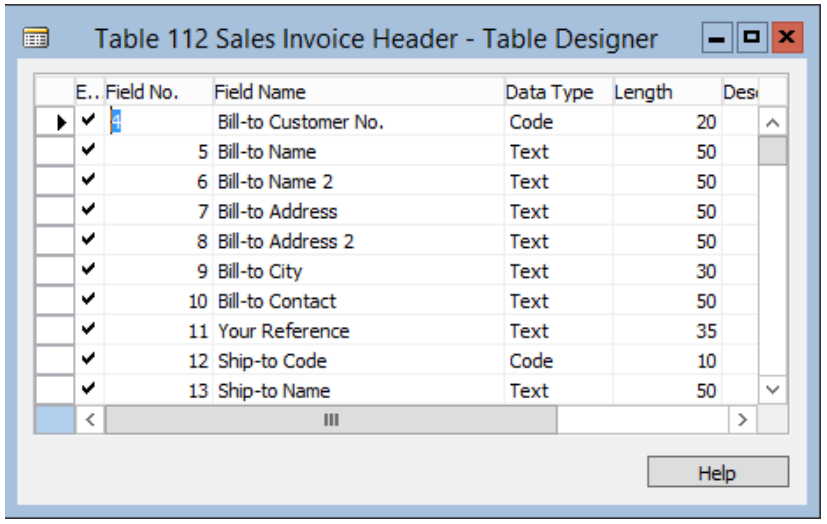
```
TransfldsFromSalesToPurchLine(VAR FromSalesLine : Record "Sales Line";VAR ToPurchLine : Record "Purchase Line")
WITH ToPurchLine DO BEGIN
  VALIDATE(Type,FromSalesLine.Type);
  VALIDATE("No.",FromSalesLine."No.");
  VALIDATE("Variant Code",FromSalesLine."Variant Code");
  VALIDATE("Location Code",FromSalesLine."Location Code");
  VALIDATE("Unit of Measure Code",FromSalesLine."Unit of Measure Code");
  IF (Type = Type::Item) AND ("No." <> '') THEN
    UpdateUOMQtyPerStockQty;
  "Expected Receipt Date" := FromSalesLine."Shipment Date";
  "Bin Code" := FromSalesLine."Bin Code";
  IF (FromSalesLine."Document Type" = FromSalesLine."Document Type"::"Return Order") AND
     ("Document Type" = "Document Type"::"Return Order")
  THEN
    VALIDATE(Quantity,FromSalesLine.Quantity)
  ELSE
    VALIDATE(Quantity,FromSalesLine."Outstanding Quantity");
  VALIDATE("Return Reason Code",FromSalesLine."Return Reason Code");
  VALIDATE("Direct Unit Cost");
  Description := FromSalesLine.Description;
  "Description 2" := FromSalesLine."Description 2";
END;
```

Copy Document

- Consistent implementation of fields
- To get the TRANSFERFIELDS to work

```
FromSalesHeader.TRANSFERFIELDS(FromSalesInvHeader);
```





SELECT DISTINCT using Queries

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable - User Friendly	WHO is part of the project?	WHAT next?
		NAV cookbook: - Cached Web Service Calls - Copy Document - SELECT DISTINCT using Queries		

SELECT DISTINCT using Queries

Challenge

- SELECT DISTINCT (a.k.a. select unique) is not provided by NAV out of the box

Approach

- Use a query object to
  - configure grouping parameters for SELECT DISTINCT
  - add a Totals column on the query to trigger grouping

SELECT DISTINCT using Queries

Example

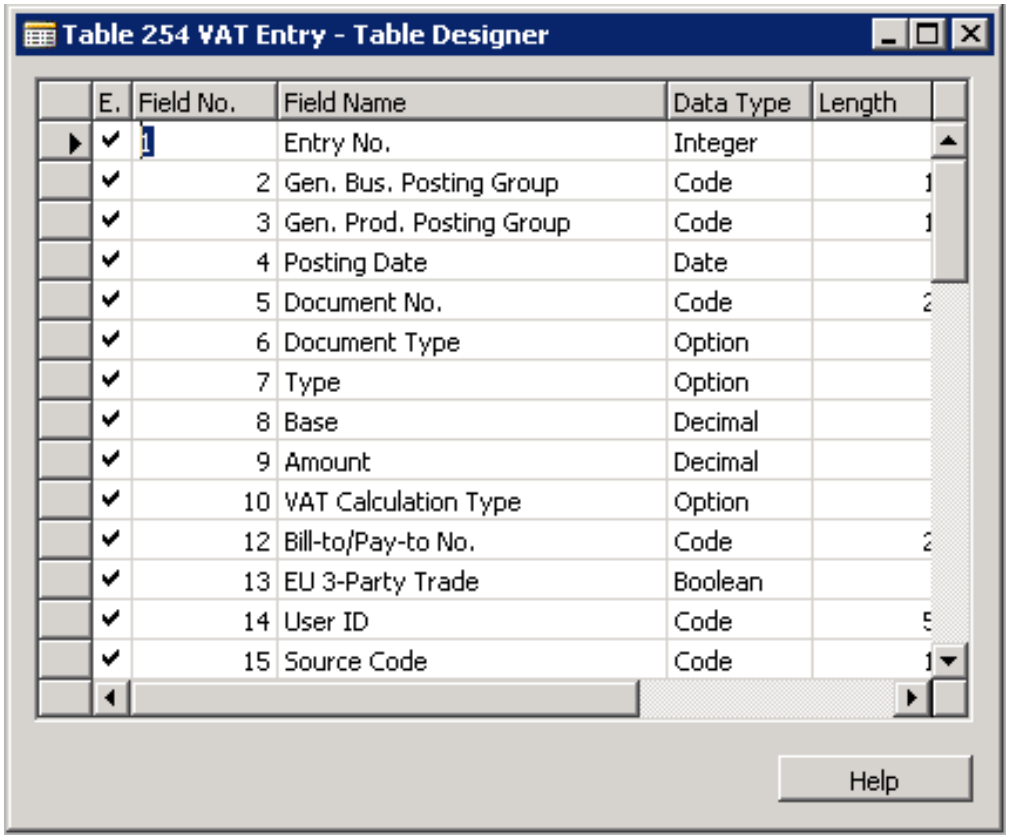
VAT Entry – get all documents which generated entries in the VAT Entry table.

DISTINCT: One line per one document.

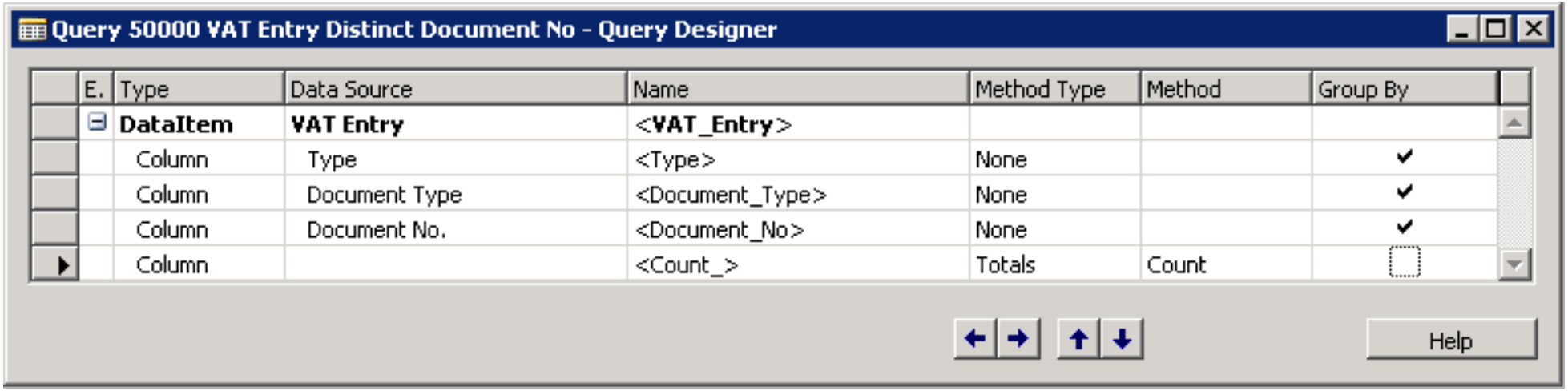
Type

Document Type

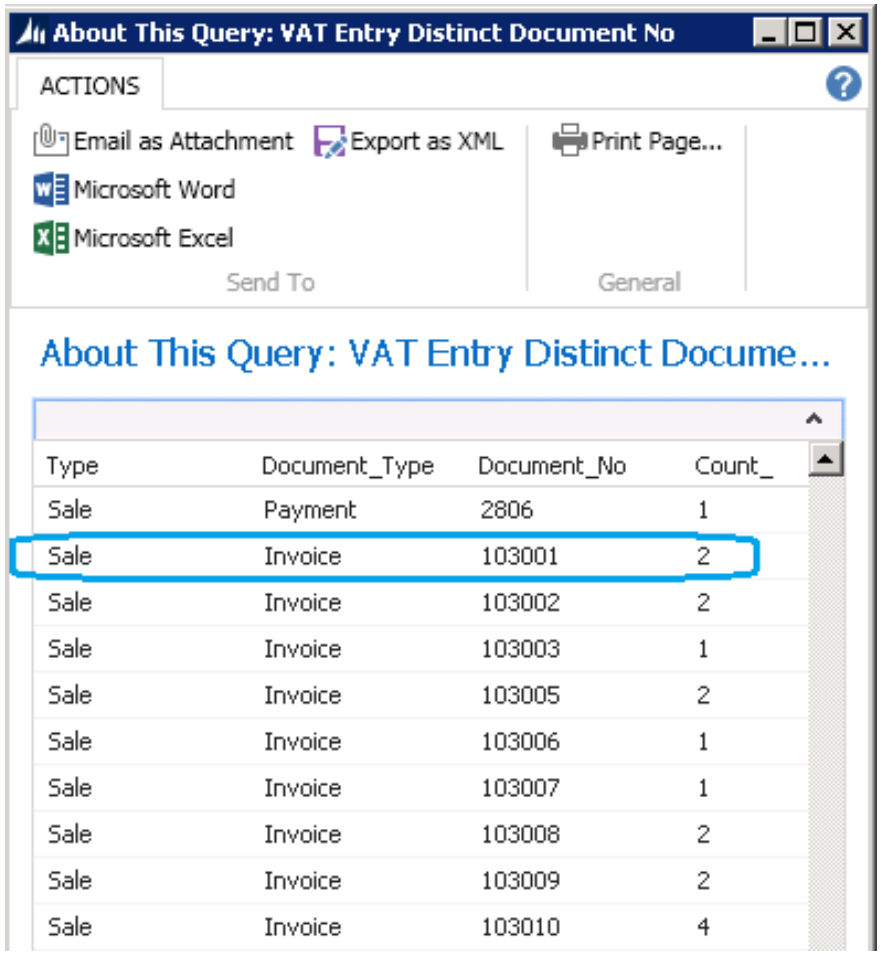
Document



SELECT DISTINCT using Queries



SELECT DISTINCT using Queries



SELECT DISTINCT using Queries

NAV Usages

Query 19: **VAT Entries Base Amt. Sum**

Used by Report 19: **VAT-VIES Declaration Tax Auth.**

Related pattern

SELECT DISTINCT with temporary tables

C/AL Guidelines

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting? Make your product - Extensible - Upgradable - Configurable - Adaptable - User Friendly	WHO is part of the project?	WHAT next?
		NAV cookbook: - Cached Web Service Calls - Copy Document - SELECT DISTINCT using Queries - C/AL guidelines		

C/AL Guidelines

We will take a few of them here ☺

- Parameters rule 1
- Parameters rule 2
- WITH scope
- Cyclomatic complexity (CC)

C/AL Guidelines

Parameters rule 1

The number of parameters passed to a string must match the placeholders.

Bad

```
HelloWorldMsg@1005 :
TextConst 'ENU=Hello, World!';

MESSAGE(HellowWorldMsg,TABLECAPTION);
```

Good

```
HelloWorldMsg@1005 :
TextConst 'ENU=Hello, World!';

MESSAGE(HelloWorldMsg);
```

C/AL Guidelines

Parameters rule 2

Do not declare parameters by reference if their values are not intended to be changed.

Bad

```
LOCAL PROCEDURE
ShowMessage@15(VAR Text@1000 : Text[250]);
  BEGIN
    IF (Text <> '') AND GenJnlLineInserted THEN
      MESSAGE(Text);
  END;
```

Good

```
LOCAL PROCEDURE
ShowMessage@15(Text@1000 : Text[250]);
  BEGIN
    IF (Text <> '') AND GenJnlLineInserted THEN
      MESSAGE(Text);
  END;
```

C/AL Guidelines

WITH Scope

Do not use the WITH scope when it has a variable whose name is the same as a local variable. This can lead to wrong code assumptions.

Bad

```
PROCEDURE InsertData@1("Contract Type"@1000 :
Option...);
  BEGIN
    WITH ServiceContractHeader DO BEGIN
      DimMgt.InsertServContractDim(
        ...,"Contract Type","Contract No.",0,...);
    END;
```

Good

```
PROCEDURE InsertData@1(ContractType@1000 :
Option...);
  BEGIN
    WITH ServiceContractHeader DO BEGIN
      DimMgt.InsertServContractDim(
        ...,ContractType,"Contract No.",0,...);
    END;
```

C/AL Guidelines

Cyclomatic complexity (CC)

Cyclomatic complexity, also known as V(G) or the graph theoretic number, is probably the most widely used complexity metric in software engineering. Defined by Thomas McCabe, it's easy to understand and calculate, and it gives useful results. This metric considers the control logic in a procedure. It's a measure of structural complexity. Low complexity is desirable.

How to calculate cyclomatic complexity?

CC = Number of decisions + 1

The cyclomatic complexity of a procedure equals the number of decisions plus one. What are decisions? Decisions are caused by conditional statements. In C/AL we consider IF and Case as decision in literature called CC3

C/AL Guidelines

Cyclomatic complexity (CC)

```
    CASE PostDesRec."Information Type" OF
      PostDesRec."Information Type"::"Description and Sundries":
        IF NOT SplitAccountnumber(PostDesRec.Description) THEN
          SplitInvoicenumber(PostDesRec.Description);
      PostDesRec."Information Type"::"Account No. Balancing Account":
        SplitAccountnumber(PostDesRec.Description);
      PostDesRec."Information Type"::"Name Acct. Holder":
        Name := PostDesRec.Description;
      PostDesRec."Information Type"::"Address Acct. Holder":
        Address := PostDesRec.Description;
      PostDesRec."Information Type"::"City Acct. Holder":
        City := PostDesRec.Description;
      PostDesRec."Information Type"::"Payment Identification":
        Identification := PostDesRec.Description;
    END;
  UNTIL PostDesRec.NEXT = 0;

  IF CBGStatementlineRec."Account No." = '' THEN BEGIN
    FOR i := 1 TO 5 DO BEGIN
      IF BankaccountNo[i] <> '' THEN BEGIN
        IF CBGStatementlineRec.Credit > 0 THEN BEGIN
          IF FindAccountnumber(BankaccountNo[i],TempRec."Source Type"::Customer,CBGStatementlineRec."Account No.") THEN BEGIN
            CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Customer;
            CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
            CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
            NumberOfLinesChanged := NumberOfLinesChanged + 1;
            RecChanged := TRUE;
          END;
        END ELSE BEGIN
          IF FindAccountnumber(BankaccountNo[i],TempRec."Source Type"::Vendor,CBGStatementlineRec."Account No.") THEN BEGIN
            CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Vendor;
            CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
            CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
            NumberOfLinesChanged := NumberOfLinesChanged + 1;
            RecChanged := TRUE;
          END;
        END;
      END;
    END;

    IF NOT RecChanged THEN BEGIN
      IF Name <> '' THEN BEGIN
        IF CBGStatementlineRec.Credit > 0 THEN BEGIN
          IF FindNAC(Name,Address,City,TempRec."Source Type"::Customer,CBGStatementlineRec."Account No.") THEN BEGIN
            CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Customer;
            CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
            CBGStatementlineRec."Reconciliation Status" := CBGStatementlineRec."Reconciliation Status"::Changed;
            NumberOfLinesChanged := NumberOfLinesChanged + 1;
            RecChanged := TRUE;
          END;
        END ELSE BEGIN
          IF FindNAC(Name,Address,City,TempRec."Source Type"::Vendor,CBGStatementlineRec."Account No.") THEN BEGIN
            CBGStatementlineRec."Account Type" := CBGStatementlineRec."Account Type"::Vendor;
            CBGStatementlineRec.VALIDATE("Account No.",CBGStatementlineRec."Account No.");
```

WHO works on design patterns?

WHAT is a Design Pattern?	WHY NAV Patterns?	WHAT NAV Patterns are we documenting?	**WHO is part of the project?**	WHAT next?

WHO is in?

Microsoft

Anders Larsen
Bardur Knudsen
Bogdan Sturzoiu
Bogdana Botez
Ciprian Iordache
Eva Dupont
Kurt Juvyns
Michael Nielsen
Mostafa Balat
Nikola Kukrika

Partners

Arend-Jan Kauffmann, Xperit Products
Claus Lundstrøm
Eric Wauters, iFacto, PRS
Gerhard Winter, agiles, PRS
Henrik Langbak, Bording Data A/S
Jan Hoek, IDYN
Kim Ginnerup, Bording Data A/S
Luc Van Vugt, fluxxus.nl
Mark Brummel, Brummel Dynamics Services, PRS
Mike Doster, Mergetool
Søren Klemmensen, Concept Computer Corp.
Xavier Garonnat, knk Ingenerie

Give us feedback

Join us

Search for “NAV Design Patterns” on

msdn Microsoft Developer Network

Microsoft Dynamics Partner Source

Find us on: facebook

LinkedIn

YouTube

Microsoft Dynamics Microsoft Dynamics Community

Design Patterns Wiki

Table of Contents

NAV C/AL Design Patterns

Find here a repository of NAV C/AL designs and examples, from generic design patterns to NAV specific howtos and cookbook recipes.
They are explained by C/AL developers who work with NAV code daily, either at Microsoft, or in the partner community.
Feel free to add you own - drop a few lines to Bogdana Botez describing your idea.

Hoping this work will bring value to you,

The NAV Design Patterns team.

WHO works on design patterns?

WHAT is a Design Pattern?	**WHY NAV Patterns?**	**WHAT NAV Patterns are we documenting?**	**WHO is part of the project?**	**WHAT next?**

Plans

- A new Pattern per month (12-15 over the next year)
- Coming up soon
  - The C/AL guidelines on the Wiki

Think NAV
in design patterns.

Any Questions?

THANK YOU

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | *www.navtechdays.com*

NAV TECH DAYS 2014

mibuso.com

COFFEE BREAK

see you back in **30** min.