

NAV  
TECH  
DAYS  
2014

mibuso.com

# HOW TO WRITE REPEATABLE SOFTWARE

Mark Brummel, Søren Klemmensen, Gary Winter, Vjeko Babic  
(Partner Ready Software)

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)

NAV  
TECH  
DAYS  
2014

mibuso.com



# INTRODUCTION

Vjekoslav  
Babic

CROATIA

Fortempo - Business  
Consulting



Mark  
Brummel

NETHERLANDS

Brummel  
Automatisering



Soren  
Klemmensen

CANADA

Concept Computer Corp



Gary Winter

GERMANY

agiles Group



## Partner Ready Software

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)



# AGENDA

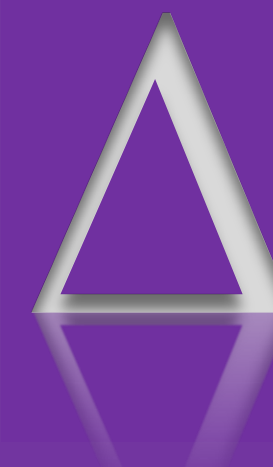
## Overview



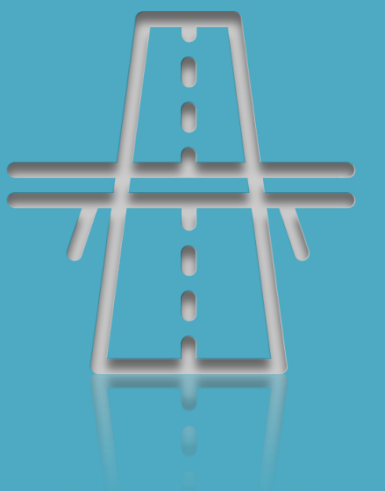
## Methodology



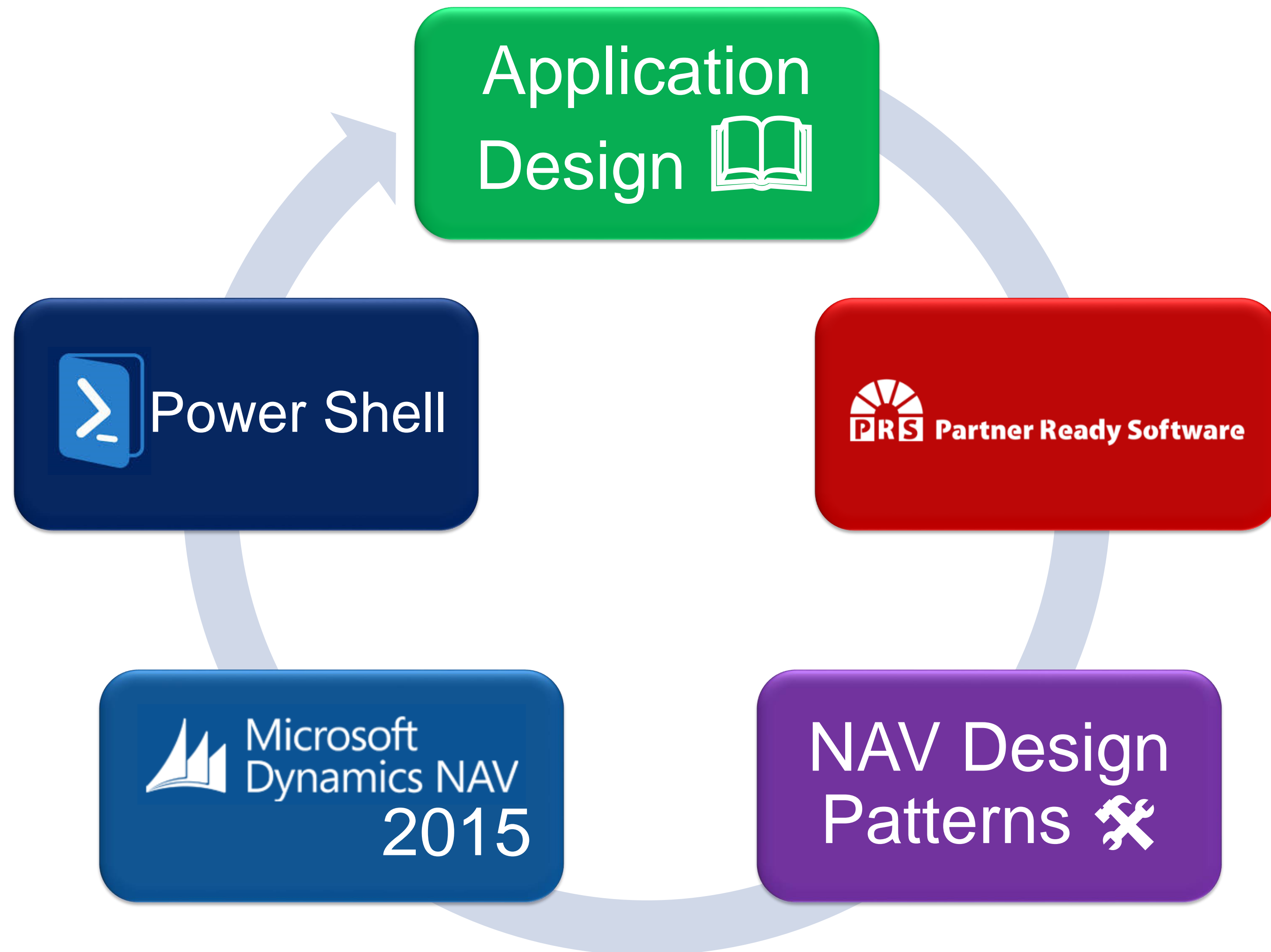
## Out of the box



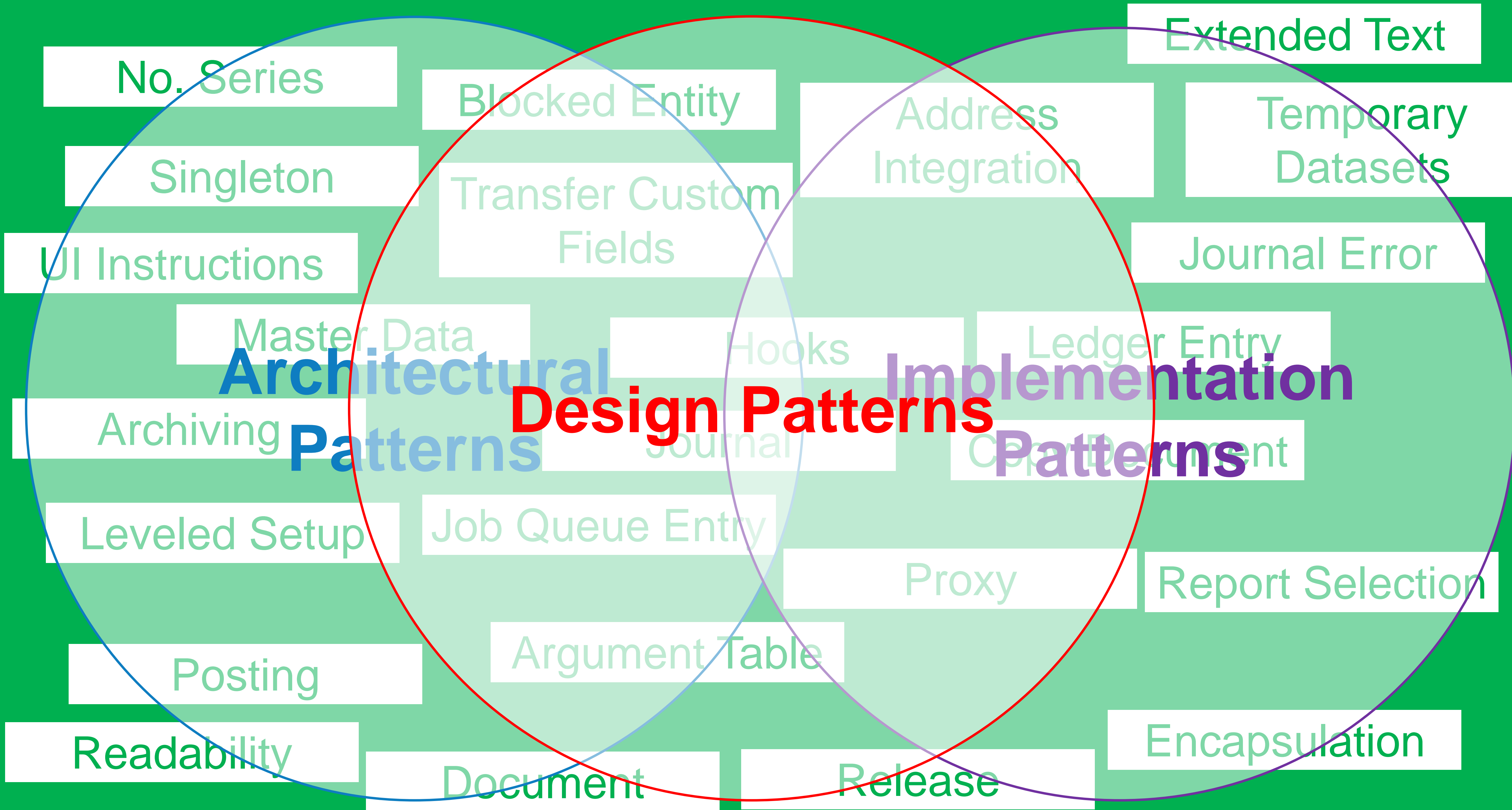
## Roadmap









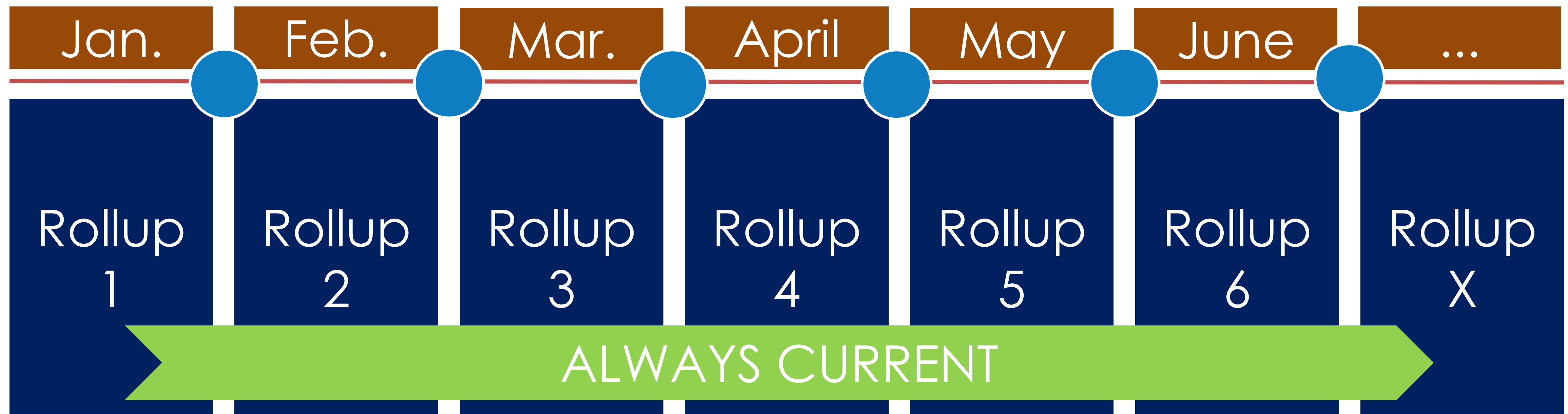


# MICROSOFT DYNAMICS NAV VERSIONS





# MICROSOFT DYNAMICS NAV ROLLUPS



# HOW TO STAY CURRENT

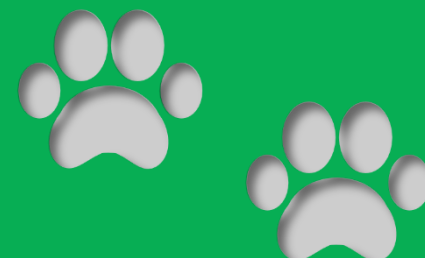
Power  
Shell



NAV  
Design  
Patterns



Reduce  
footprint



WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)

NAV  
TECH  
DAYS  
2014

[mibuso.com](http://mibuso.com)





Upgrade in Minutes

# Total Cost of Ownership





# Methodology



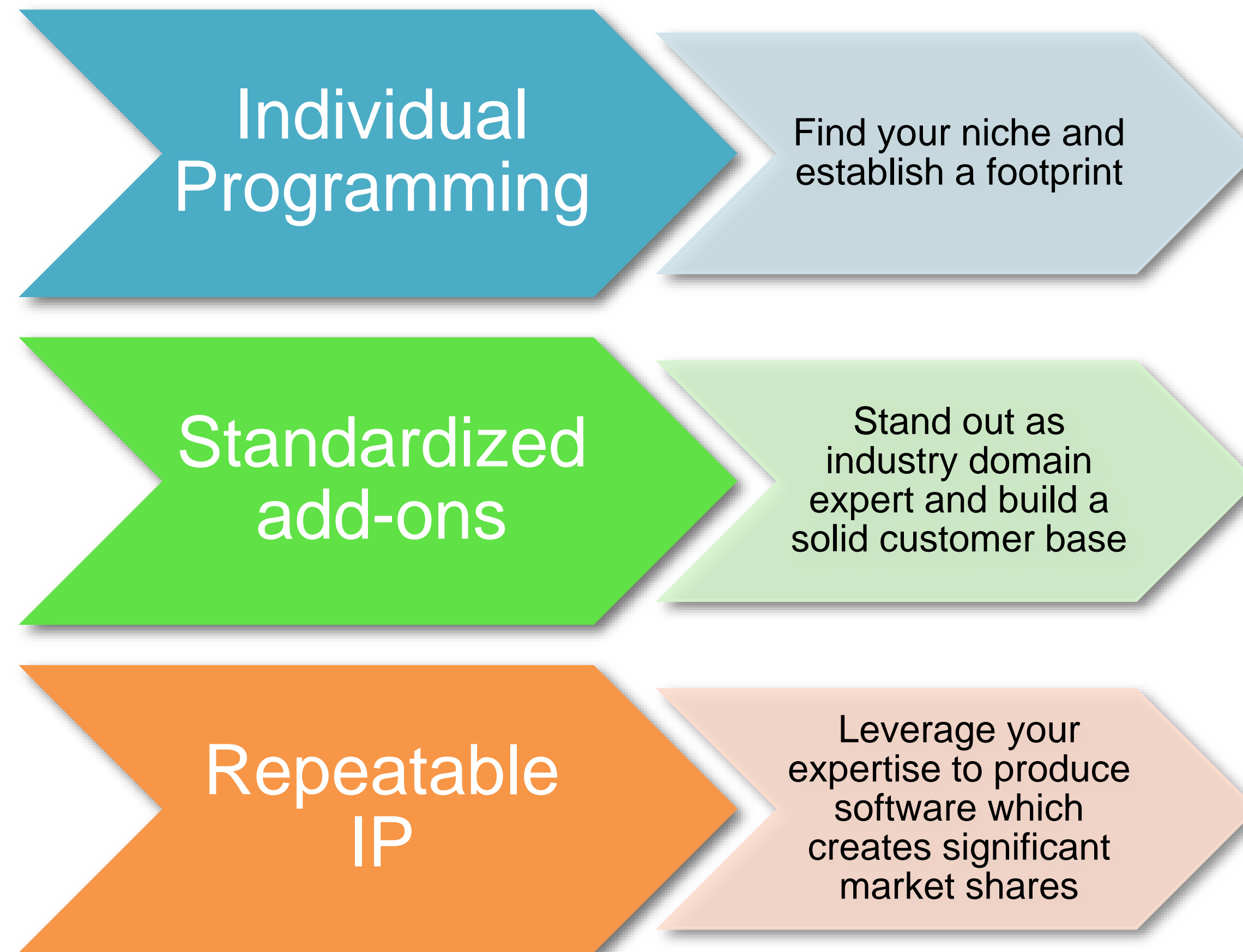
What Is  
Repeatable IP?

How Do I Write  
Repeatable IP?

How Do I Get  
Started?



# Types of Development



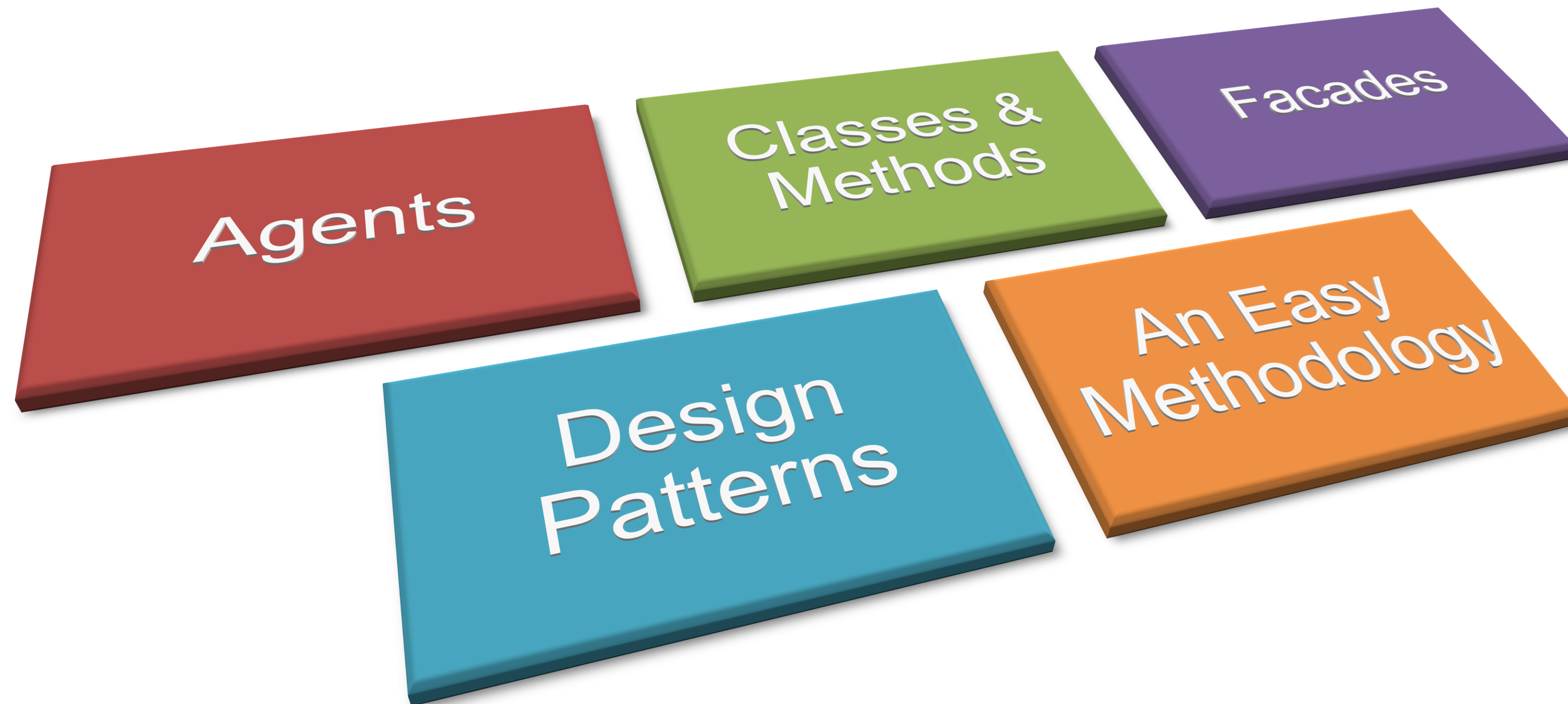
What Is  
Repeatable IP?

How Do I Write  
Repeatable IP?

How Do I Get  
Started?



# How Do I Write Code in a Repeatable Way?



# Agents

PROCESS

AGENT

ACTION

Medic

At his Office

Diagnose

On Home Visit

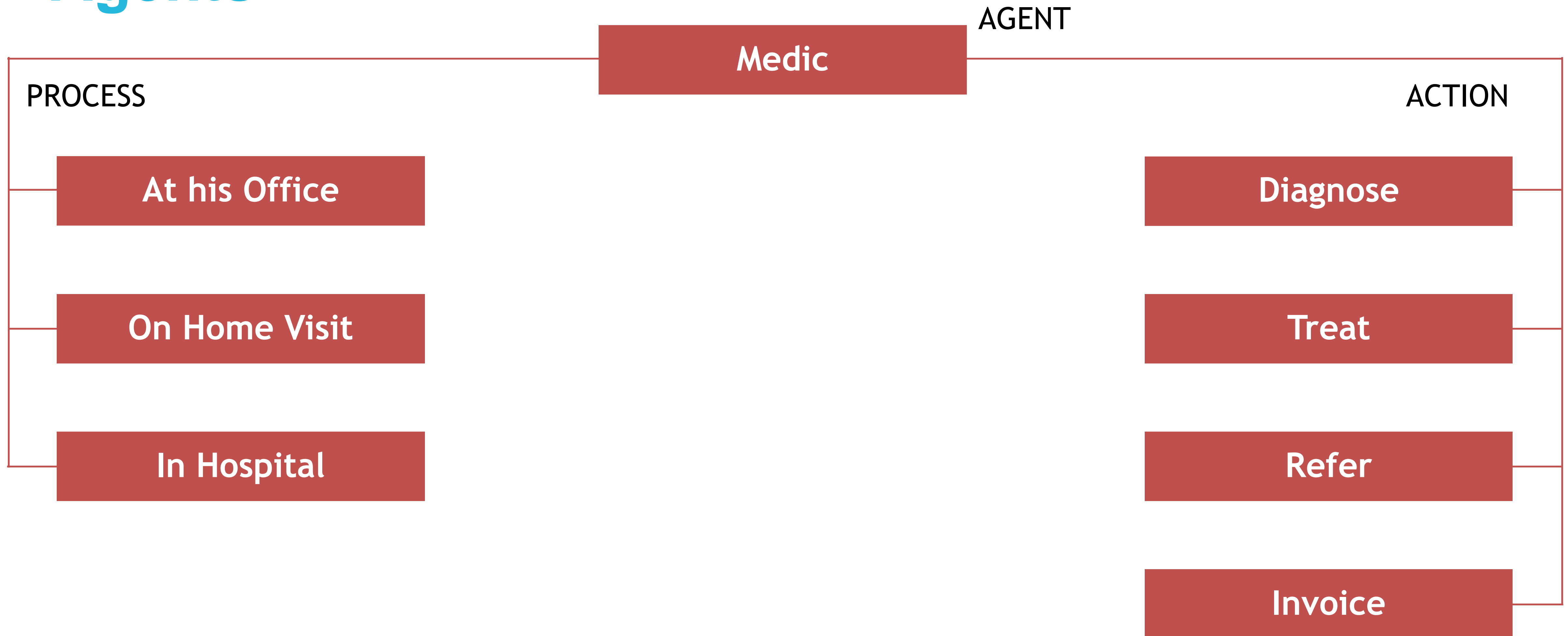
Treat

In Hospital

Refer

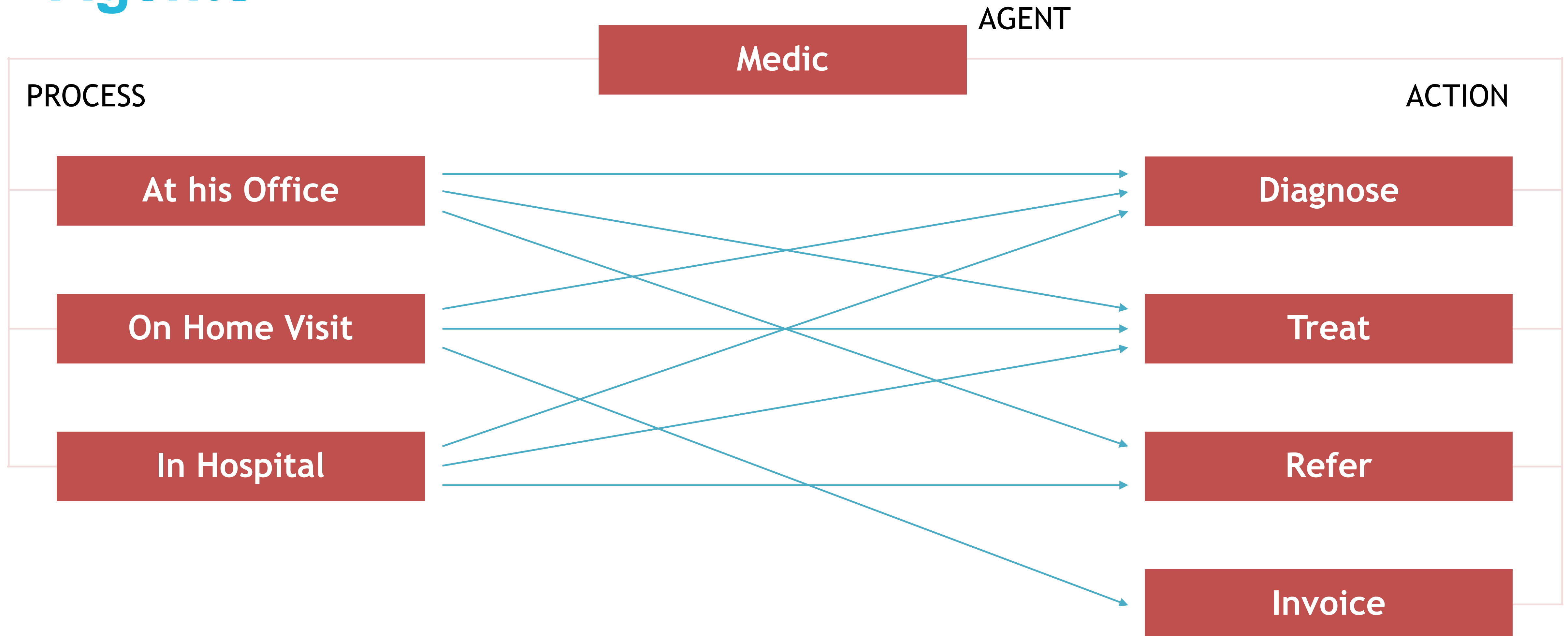
Invoice

# Agents

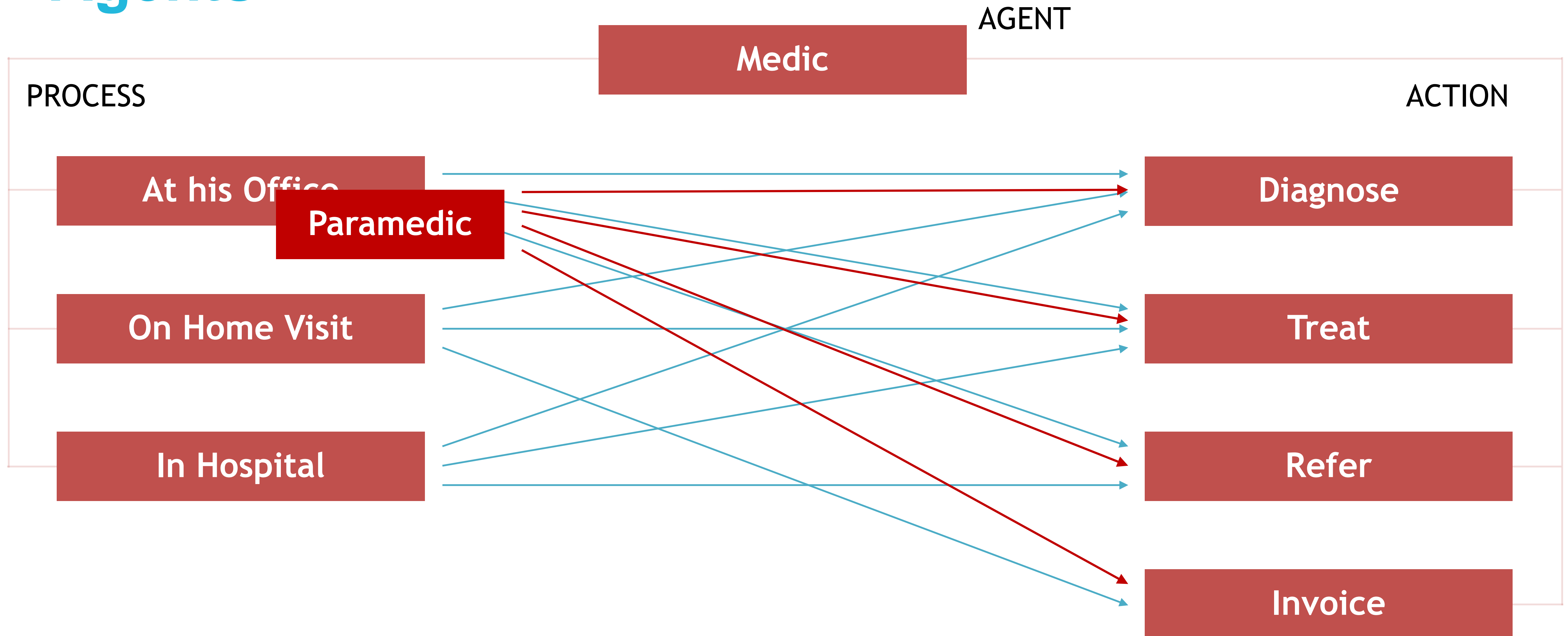




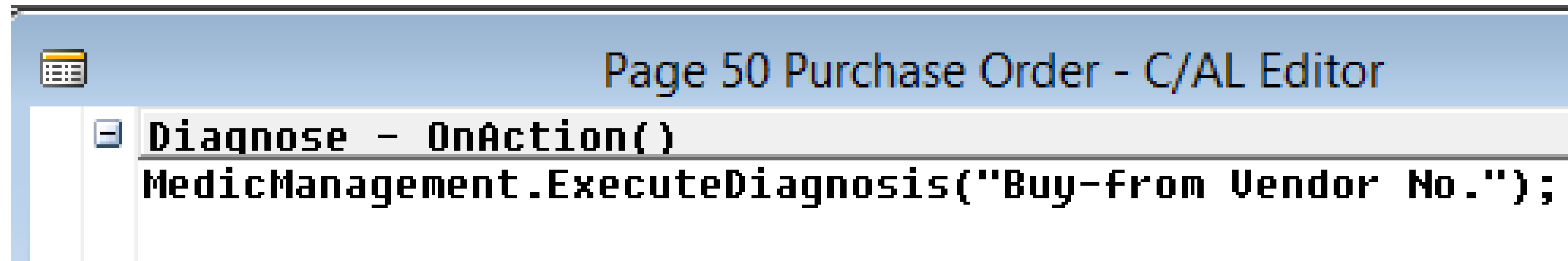
# Agents



# Agents




# Agents





# Agents



Page 50 Purchase Order - C/AL Editor

```
Diagnose - OnAction()  
MedicManagement.ExecuteDiagnosis("Buy-from Vendor No.");
```


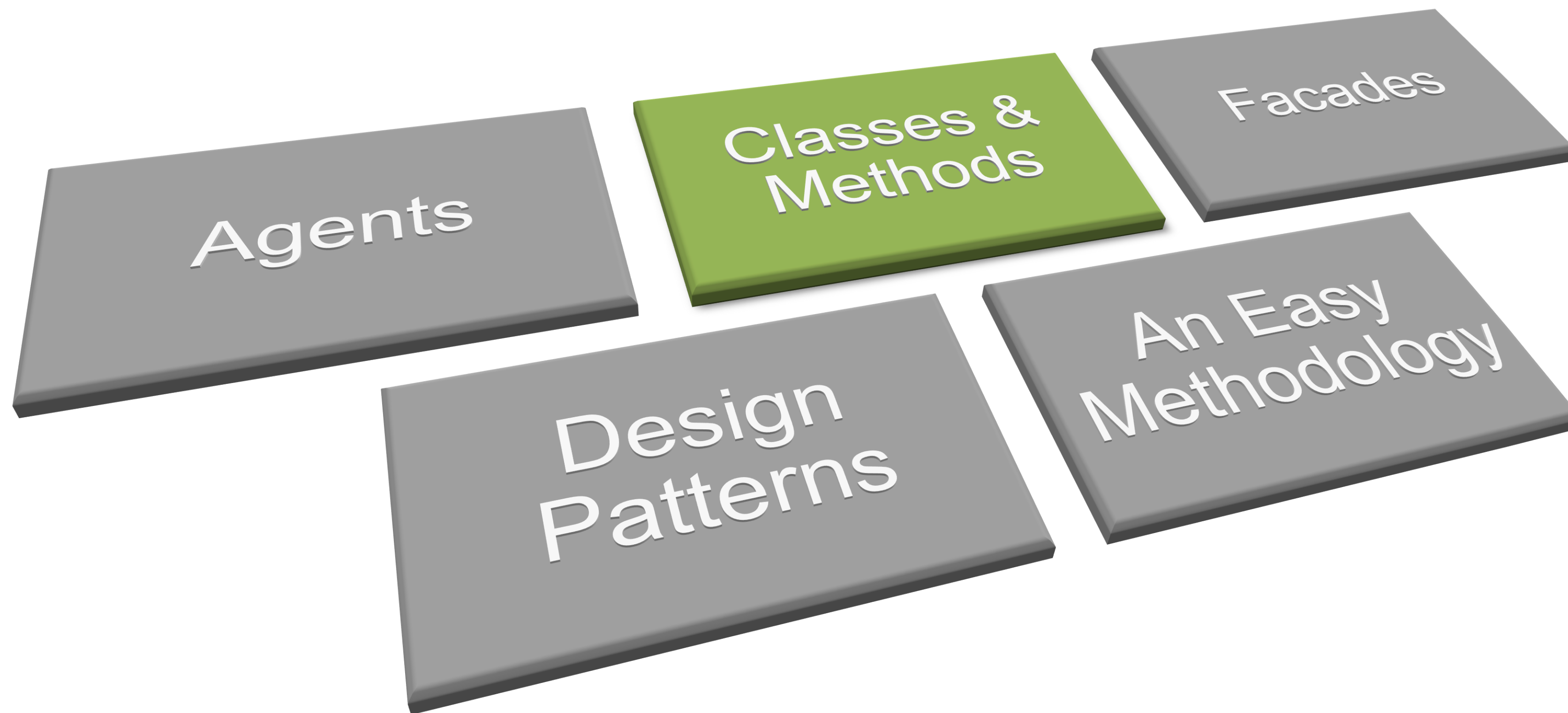


Table 38 Purchase Header - C/AL Editor

```
Diagnose()  
Medic.GET("Buy-from Vendor No.");  
Medic.Diagnose;
```

# Agents

- Translations:
  - Agent => Class => Table
  - Action => Method => Function (Codeunit)
  - (Not looking at Processes in this context)
- Start any development by by determining who is acting
- The agent is always a class (table)





# Classes & Methods

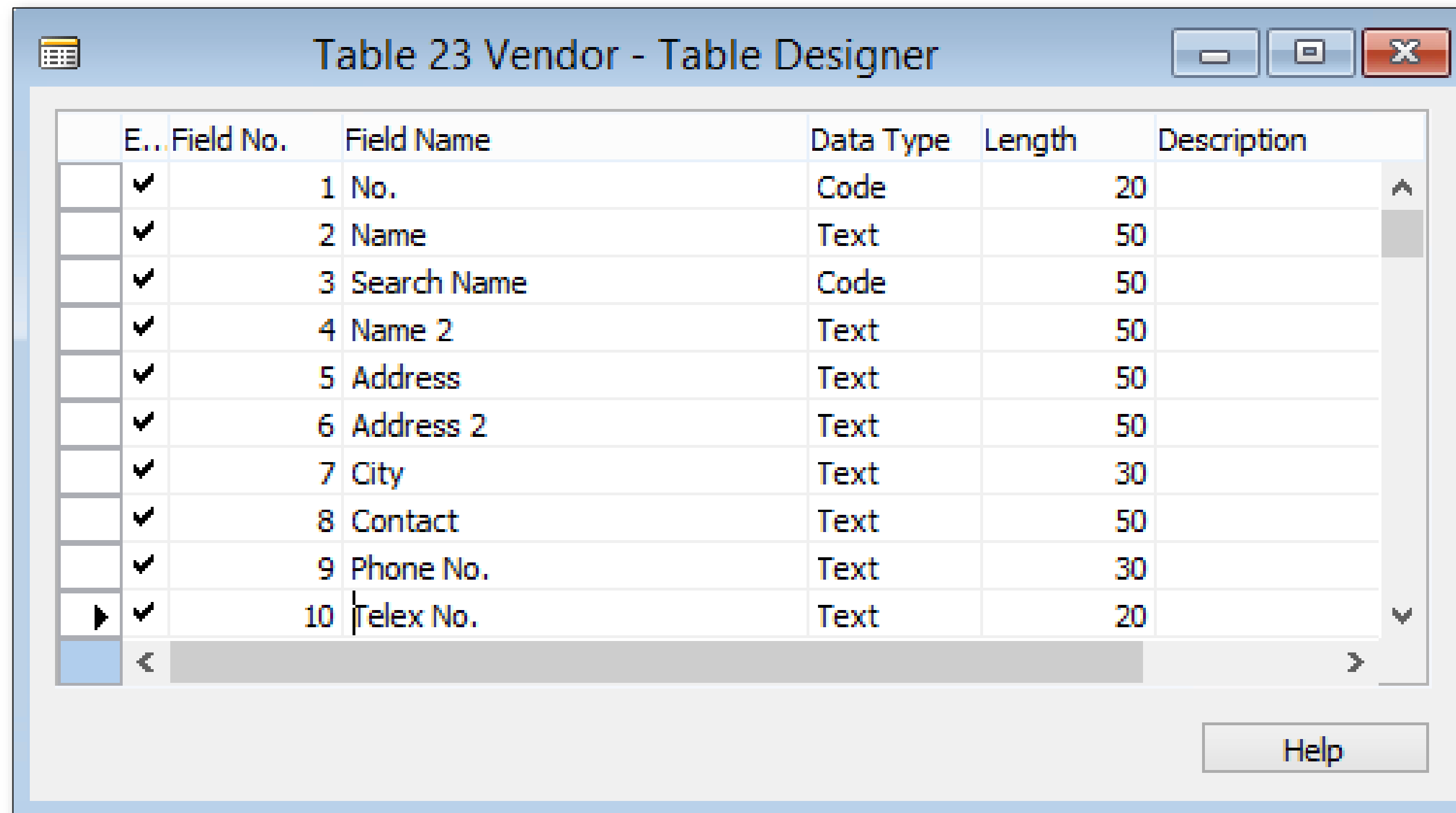
- Classes are tables
- They carry all attributes + methods
- Attributes are fields

Methods are functions

All methods are always declared on the class

# Classes & Methods

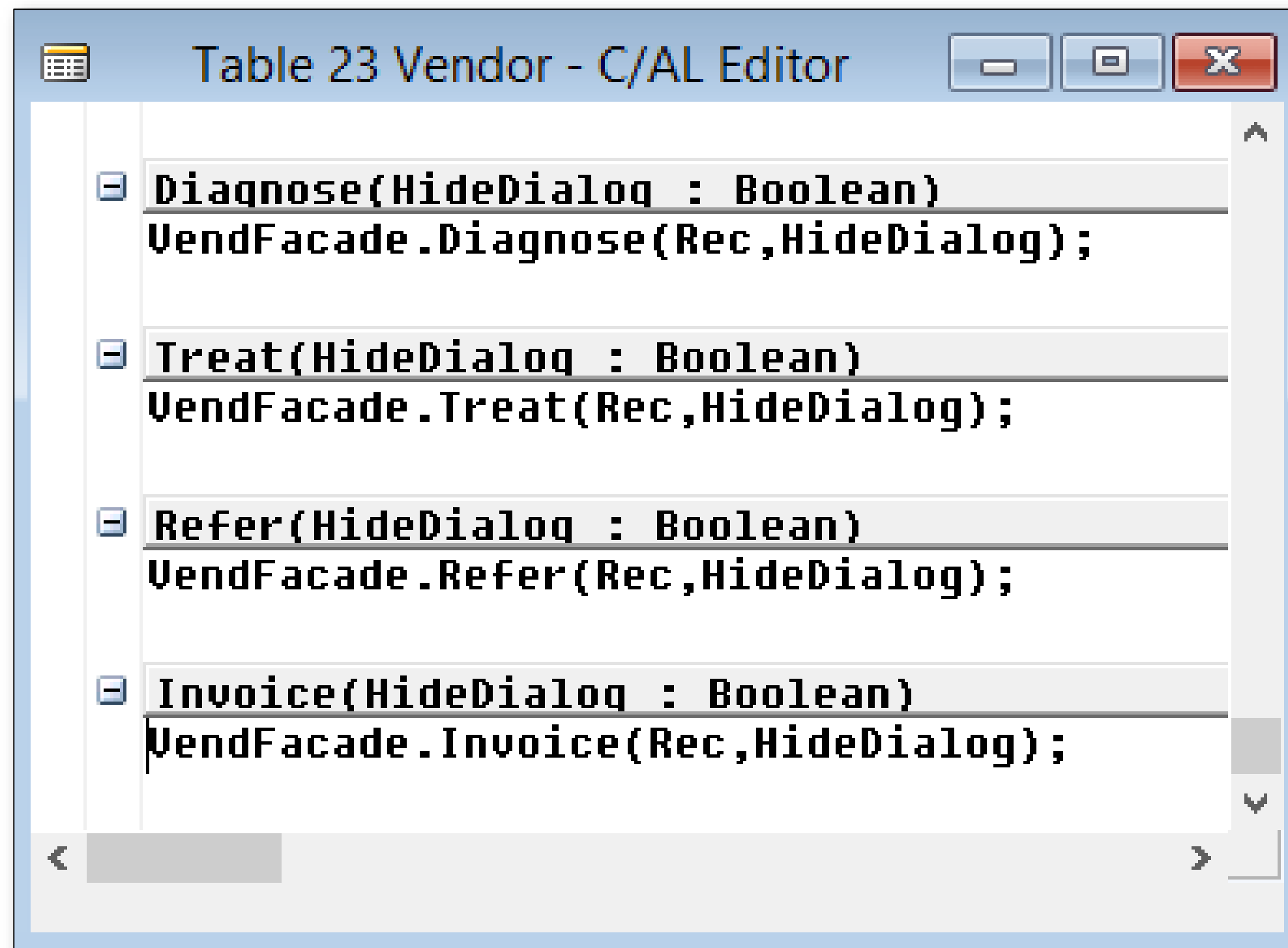
- All attributes (fields) are declared on the class (table)



E..	Field No.	Field Name	Data Type	Length	Description
✓	1	No.	Code	20	
✓	2	Name	Text	50	
✓	3	Search Name	Code	50	
✓	4	Name 2	Text	50	
✓	5	Address	Text	50	
✓	6	Address 2	Text	50	
✓	7	City	Text	30	
✓	8	Contact	Text	50	
✓	9	Phone No.	Text	30	
✓	10	Telex No.	Text	20	

# Classes & Methods

- All methods (functions) are declared on the class (table)



The screenshot shows a window titled "Table 23 Vendor - C/AL Editor". It contains four method declarations, each with a minus icon on the left:

```

Diagnose(HideDialog : Boolean)
VendFacade.Diagnose(Rec,HideDialog);

Treat(HideDialog : Boolean)
VendFacade.Treat(Rec,HideDialog);

Refer(HideDialog : Boolean)
VendFacade.Refer(Rec,HideDialog);

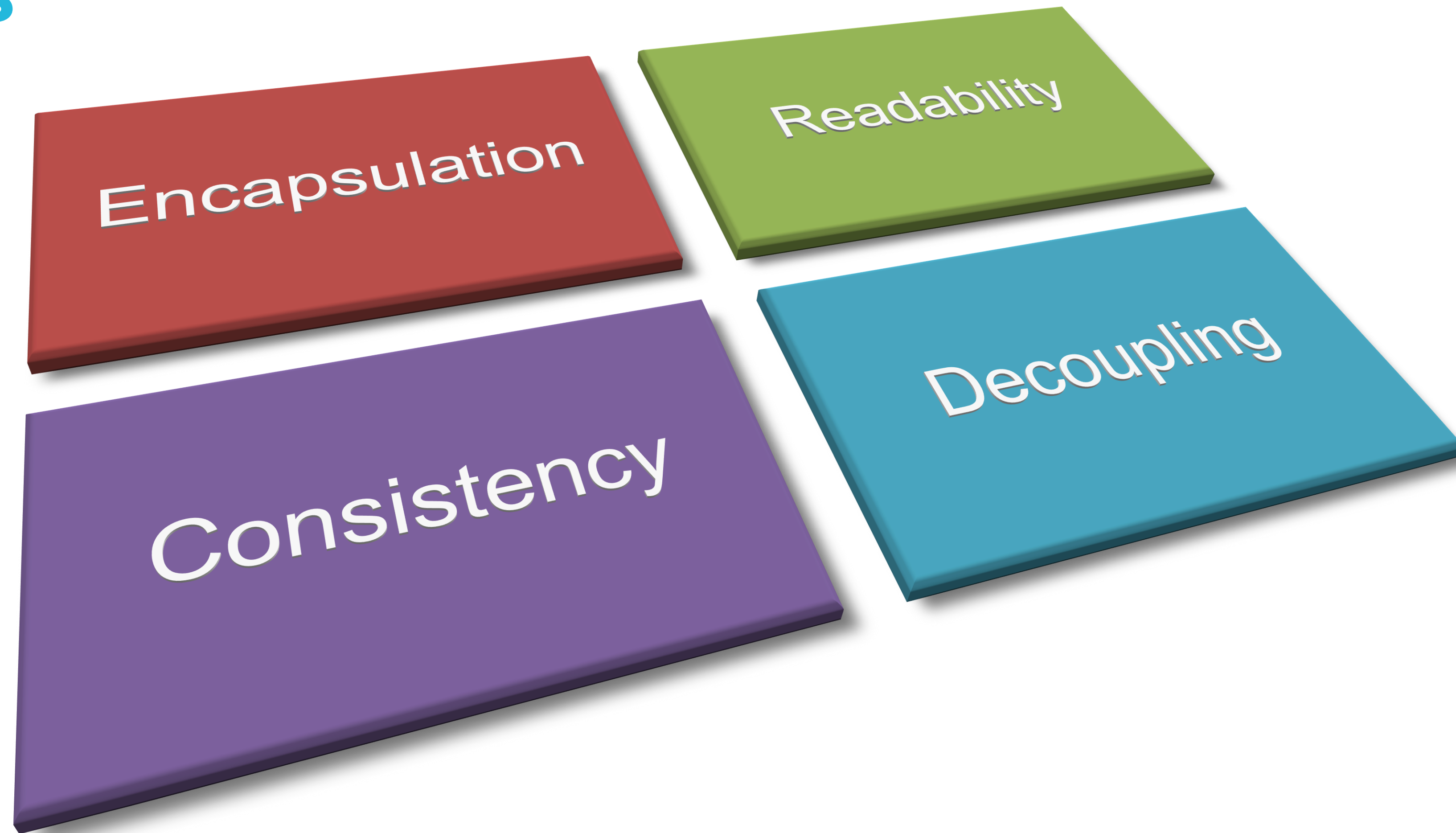
Invoice(HideDialog : Boolean)
VendFacade.Invoice(Rec,HideDialog);
  
```



# Method Declaration: Advantages

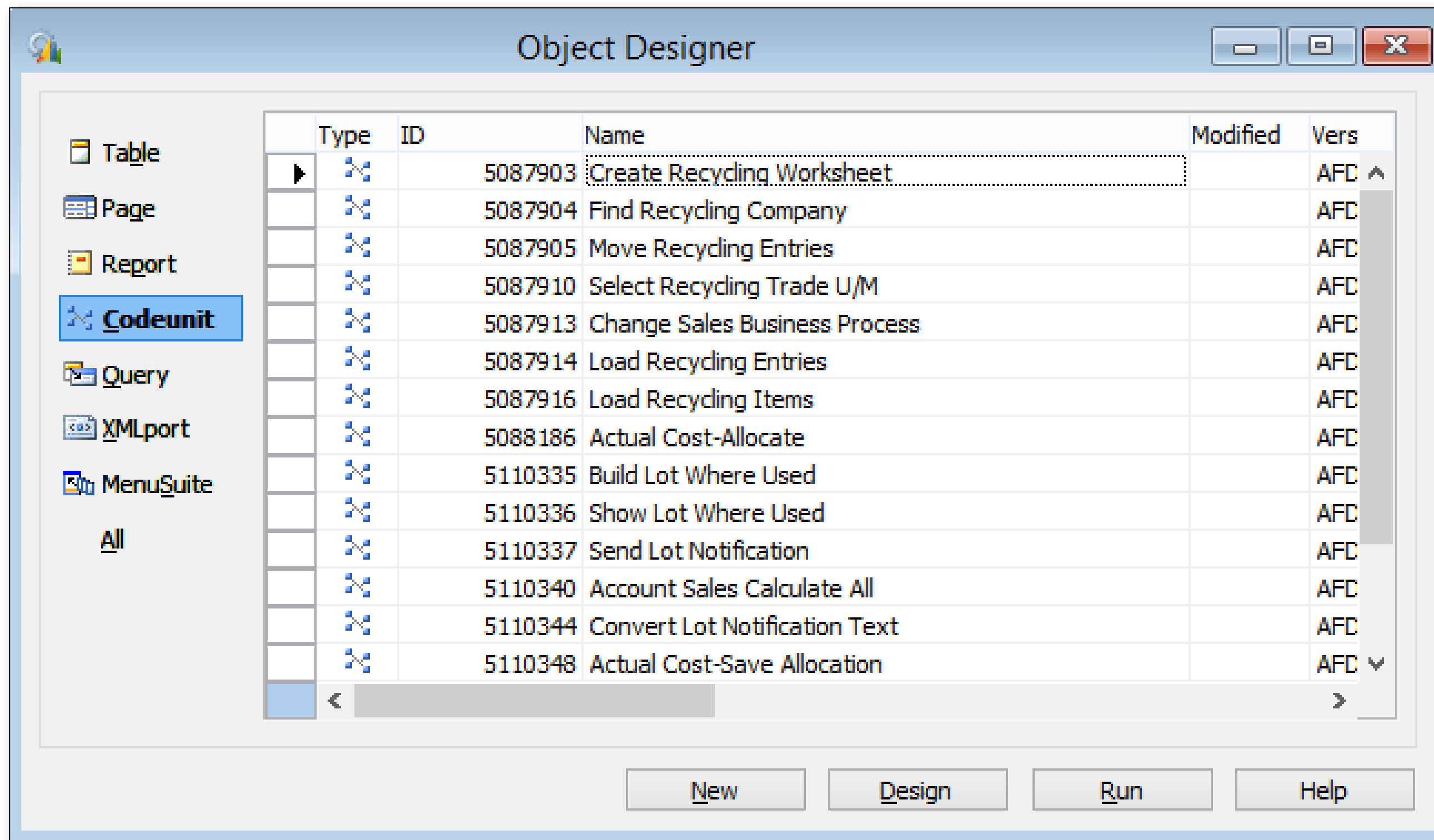
- You see explicitly ...
  - which functions exist for each table
  - which methods exist for each class
  - which actions are defined for each agent
- IntelliSense (F5)
- But ...
  - methods should not be coded on the class

# Methods



# Encapsulation

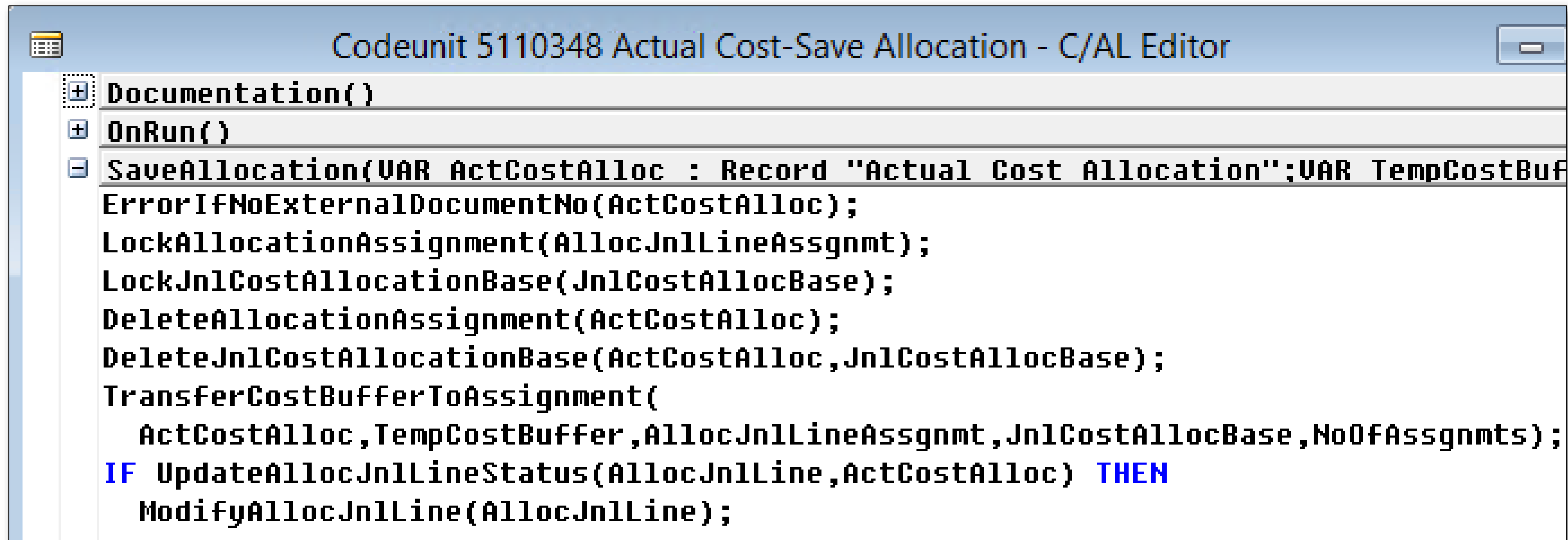
- Each method is a codeunit of its own





# Encapsulation

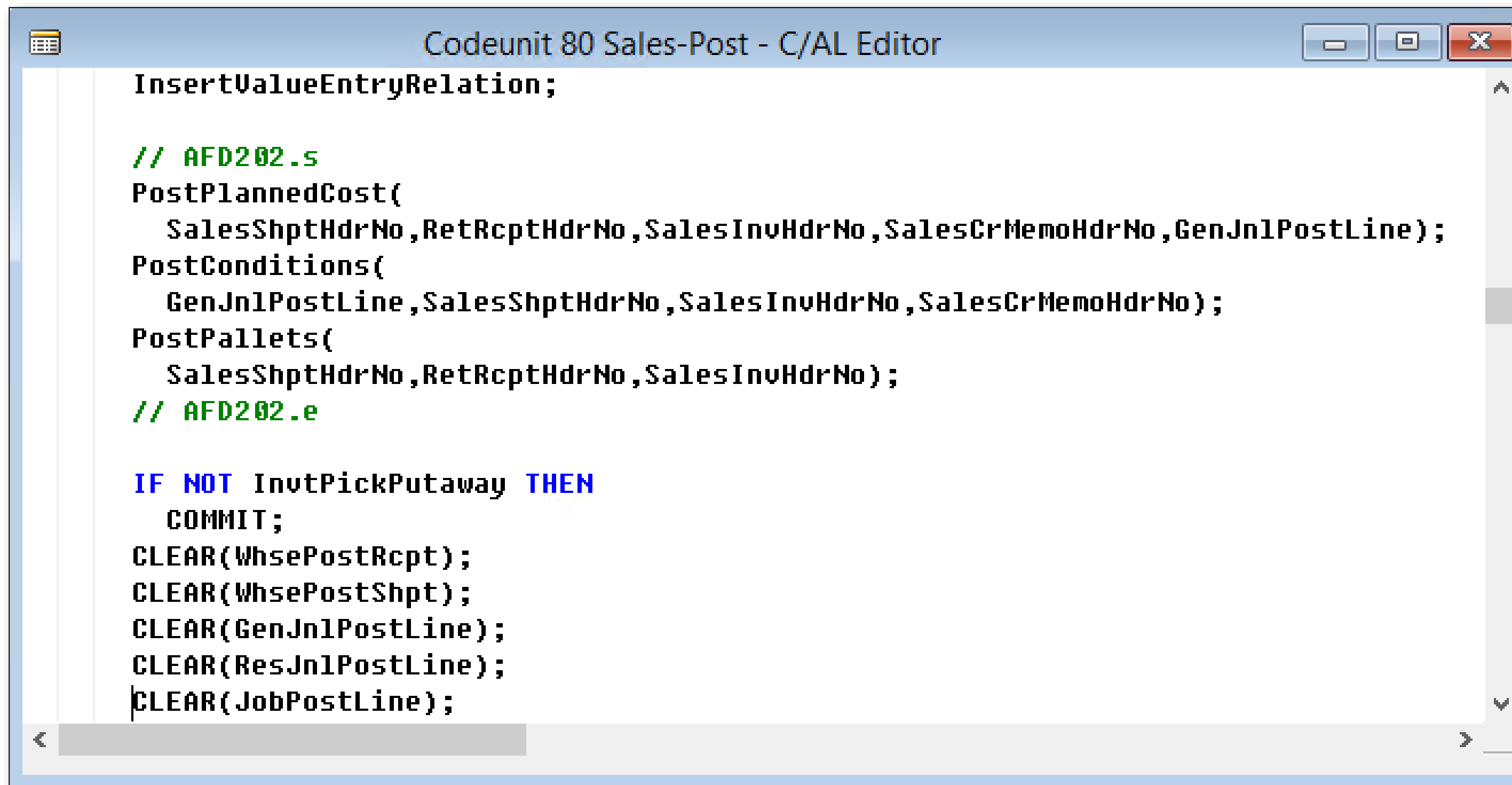
- Each method only has one global function



```
Documentation()  
OnRun()  
SaveAllocation(VAR ActCostAlloc : Record "Actual Cost Allocation";VAR TempCostBuf  
ErrorIfNoExternalDocumentNo(ActCostAlloc);  
LockAllocationAssignment(AllocJnlLineAssgnmt);  
LockJnlCostAllocationBase(JnlCostAllocBase);  
DeleteAllocationAssignment(ActCostAlloc);  
DeleteJnlCostAllocationBase(ActCostAlloc,JnlCostAllocBase);  
TransferCostBufferToAssignment(  
    ActCostAlloc,TempCostBuffer,AllocJnlLineAssgnmt,JnlCostAllocBase,NoOfAssgnmts);  
IF UpdateAllocJnlLineStatus(AllocJnlLine,ActCostAlloc) THEN  
    ModifyAllocJnlLine(AllocJnlLine);
```

# Encapsulation

- A method is always instantiated through its class



```
Codeunit 80 Sales-Post - C/AL Editor

InsertValueEntryRelation;

// AFD202.s
PostPlannedCost(
    SalesShptHdrNo,RetRcptHdrNo,SalesInvHdrNo,SalesCrMemoHdrNo,GenJnlPostLine);
PostConditions(
    GenJnlPostLine,SalesShptHdrNo,SalesInvHdrNo,SalesCrMemoHdrNo);
PostPallets(
    SalesShptHdrNo,RetRcptHdrNo,SalesInvHdrNo);
// AFD202.e

IF NOT InvtPickPutaway THEN
    COMMIT;
CLEAR(WhsePostRcpt);
CLEAR(WhsePostShpt);
CLEAR(GenJnlPostLine);
CLEAR(ResJnlPostLine);
CLEAR(JobPostLine);
```

# Method Breakdown

- Break all of your processes down into encapsulated methods
  - Get the sizing right
  - Conceptual work is mandatory (=> professional software production)
- Allows you to put these methods together again as if it was a scripting language
- Allows you to build new functionality and automated batches on existing, tested code



Encapsulation

Readability

Consistency

Decoupling

# Readability

- The global function of each method must be a readable flow chart of what the method does
- It should include all relevant steps
- Natural Language Code: It should resemble understandable plain text
- It must not contain any “nerdy” stuff
  - Cust.GET is nerdy
  - GetCustomer is not

# Readability

```
BatchPostWarehouseCost(VAR Location : Record Locat
ErrorIfNoPeriod(StartingDate,EndingDate);
ErrorIfNoPostingDate(PostingDate);
ErrorIfLocationFilterTooLong(Location);
GetCostPostingNo(CostPostingNo);
CommitTransactionToDatabase;

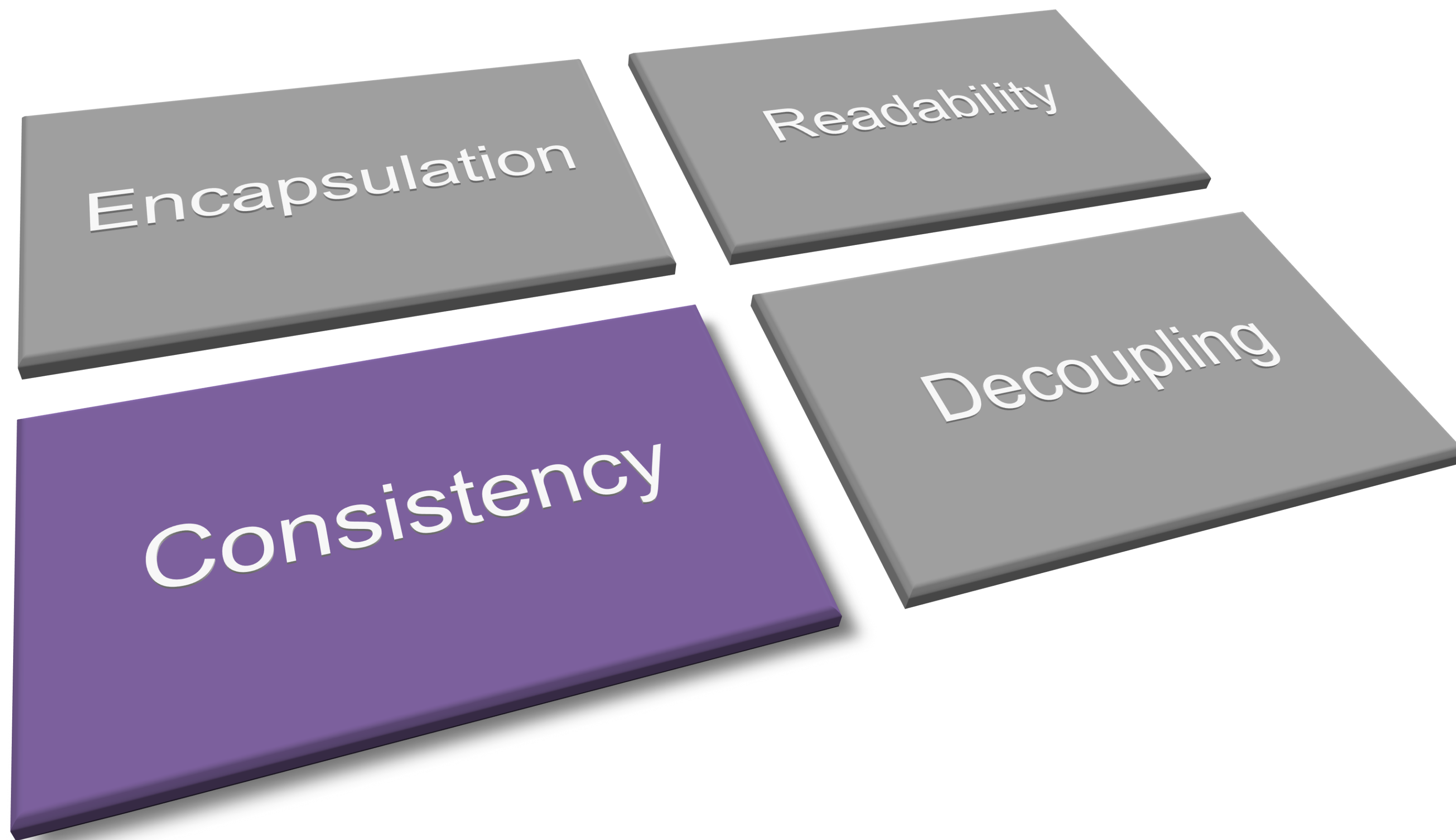
GetagilesFoodSetup(agilesFoodSetup);
GetSourceCodeSetup(SourceCodeSetup);
InitRoundingPrecision(Currency);
ClearGeneralJournalPosting(GenJnlPostLine);

ForceHideProgressIfNoUserInterface(HideProgress);
ForceNonTransitFilterOnLocation(Location);
PostWhseCostByLocation(
```



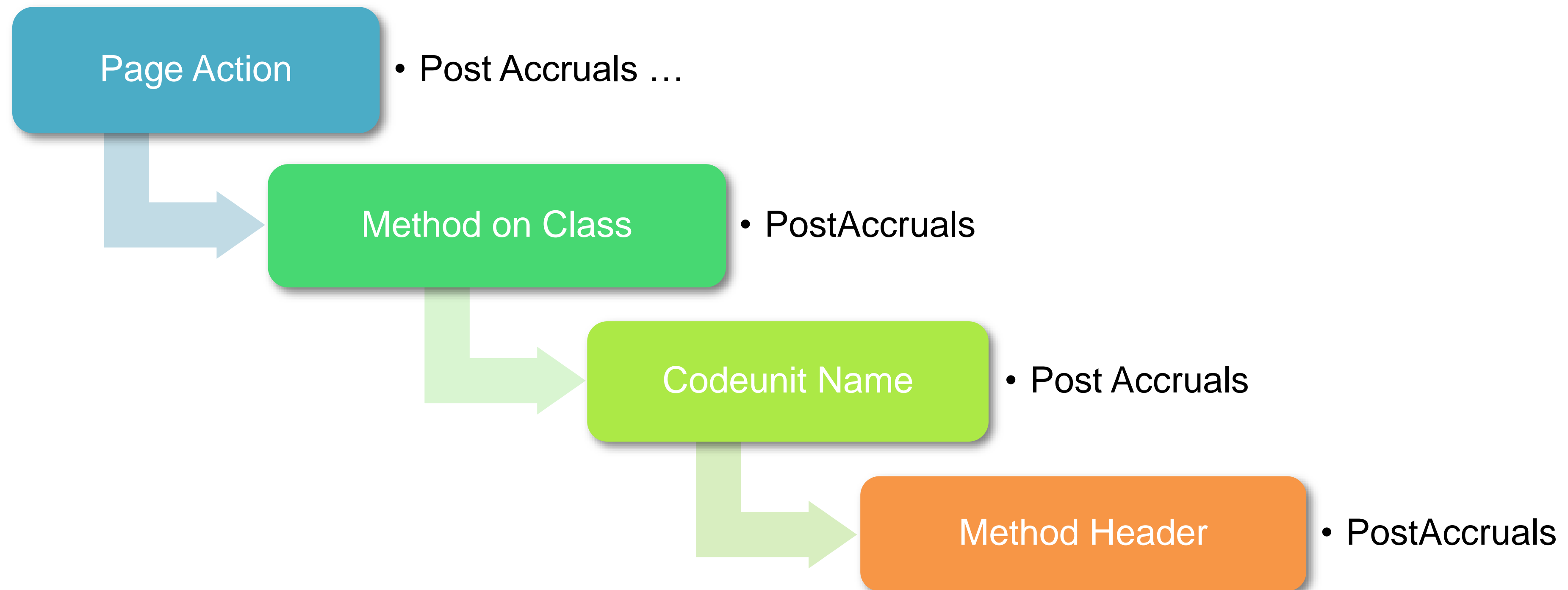
# Process Components

- Method Header
- Validity Check
  - If not valid => Exit / Error / Error Handling
- Filter Section
- Loop Section
- Initialize Section
- Database Transactions
  - Normally: Modify / Insert



# Consistency

- Each action / method follows consistent naming from UI thru Code





# Extending Consistency

- Naming should be consistent in other contexts as well
- “Posting Accruals” in:
  - Process Manual
  - Help Server
  - Consulting
  - Sales Material
  - Customer-facing Feature Description

Encapsulation

Readability

Consistency

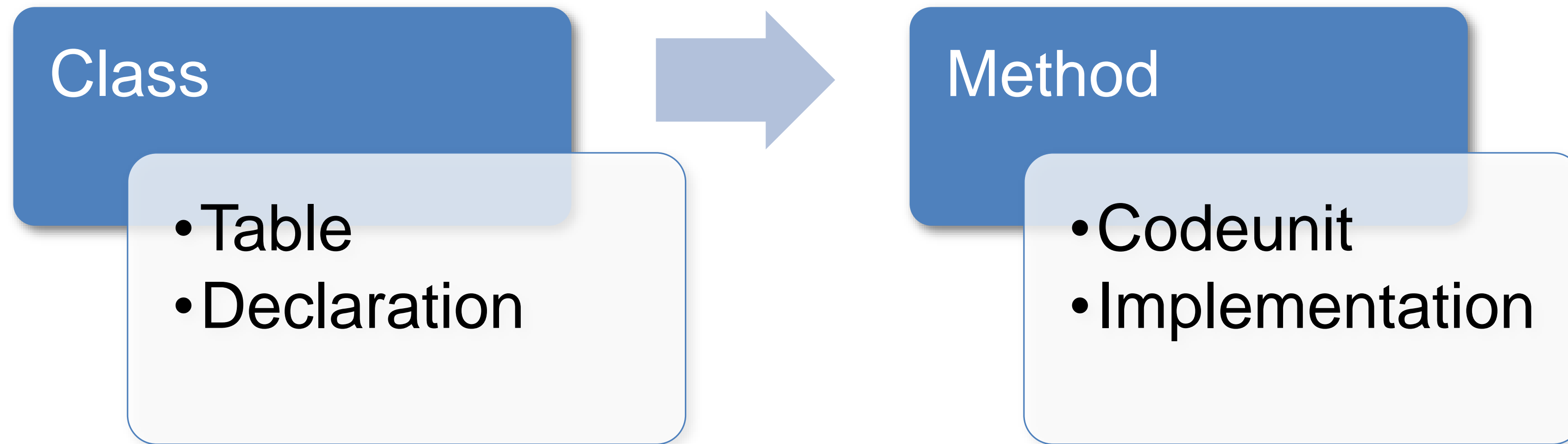
Decoupling

# Facade

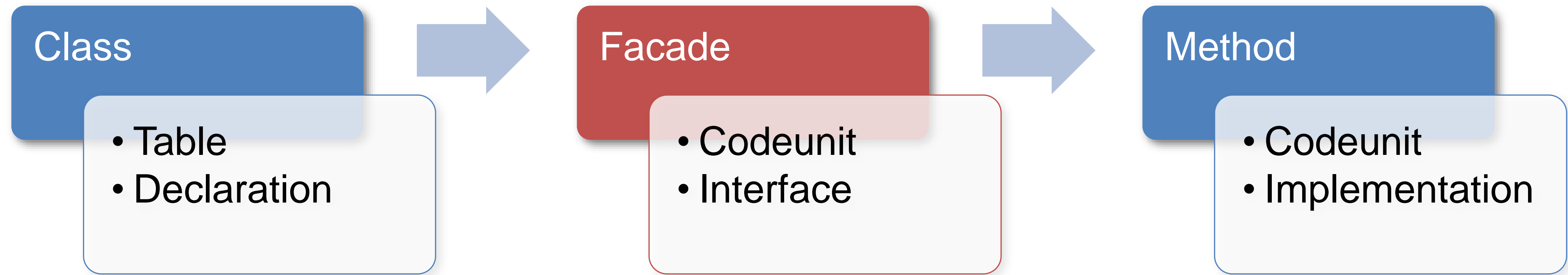




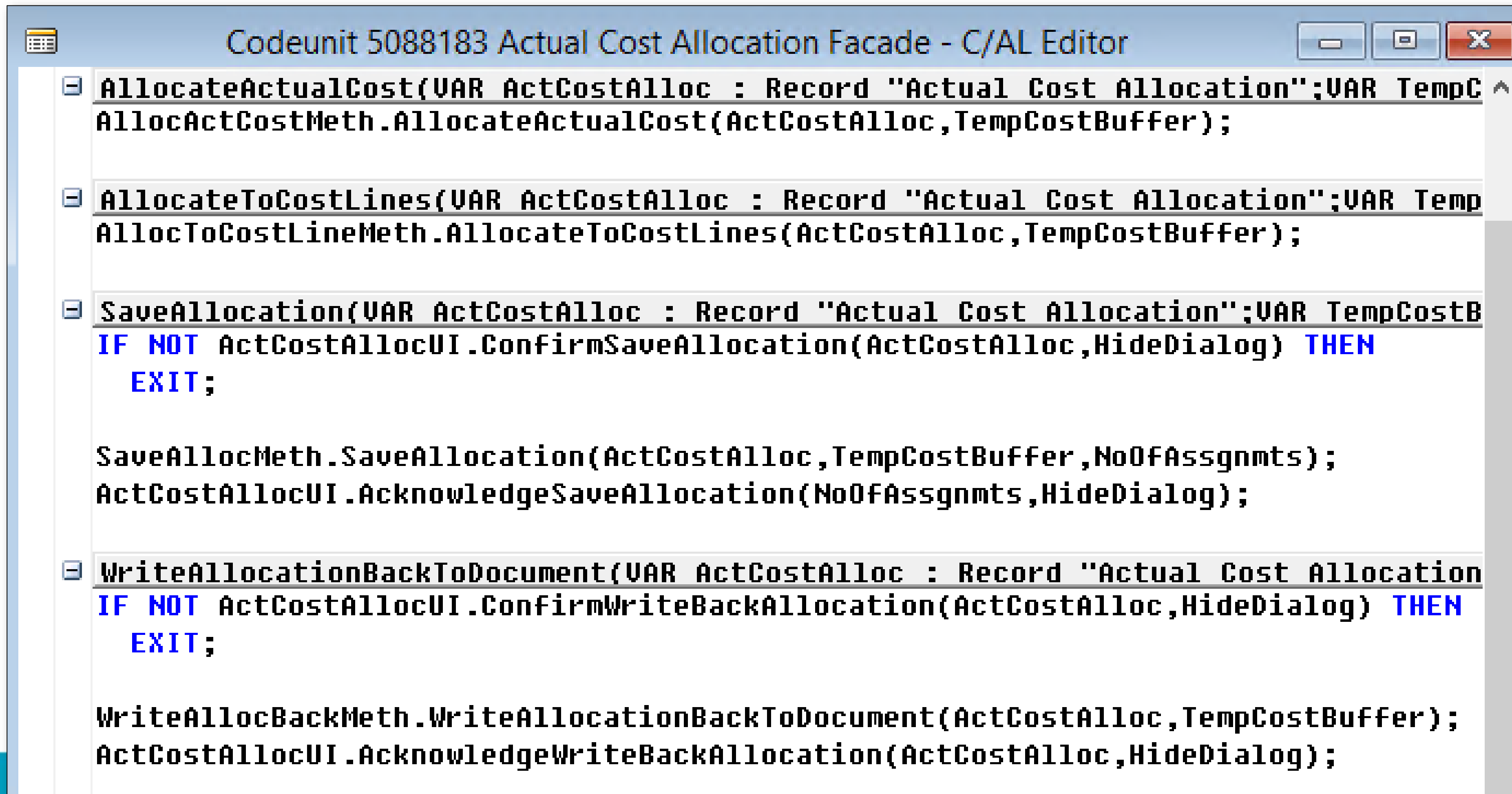
# Decoupling: Facade



# Decoupling: Facade



# Facade



```
AllocateActualCost(VAR ActCostAlloc : Record "Actual Cost Allocation";VAR TempC
AllocActCostMeth.AllocateActualCost(ActCostAlloc,TempCostBuffer);

AllocateToCostLines(VAR ActCostAlloc : Record "Actual Cost Allocation";VAR Temp
AllocToCostLineMeth.AllocateToCostLines(ActCostAlloc,TempCostBuffer);

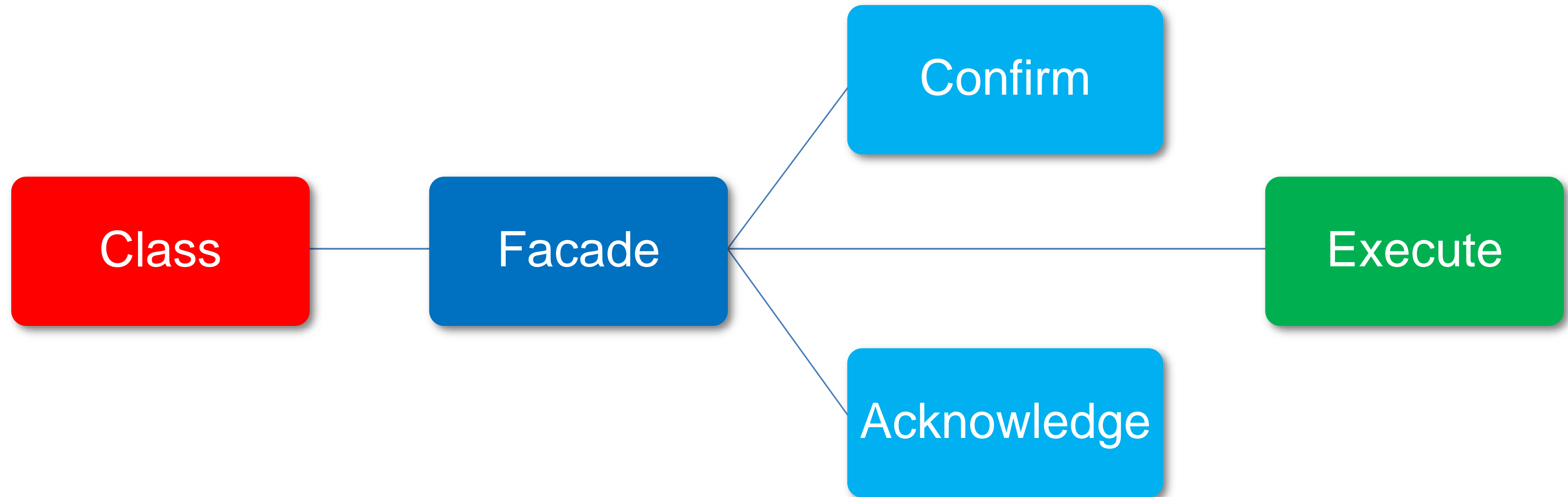
SaveAllocation(VAR ActCostAlloc : Record "Actual Cost Allocation";VAR TempCostB
IF NOT ActCostAllocUI.ConfirmSaveAllocation(ActCostAlloc,HideDialog) THEN
    EXIT;

SaveAllocMeth.SaveAllocation(ActCostAlloc,TempCostBuffer,NoOfAssgnmts);
ActCostAllocUI.AcknowledgeSaveAllocation(NoOfAssgnmts,HideDialog);

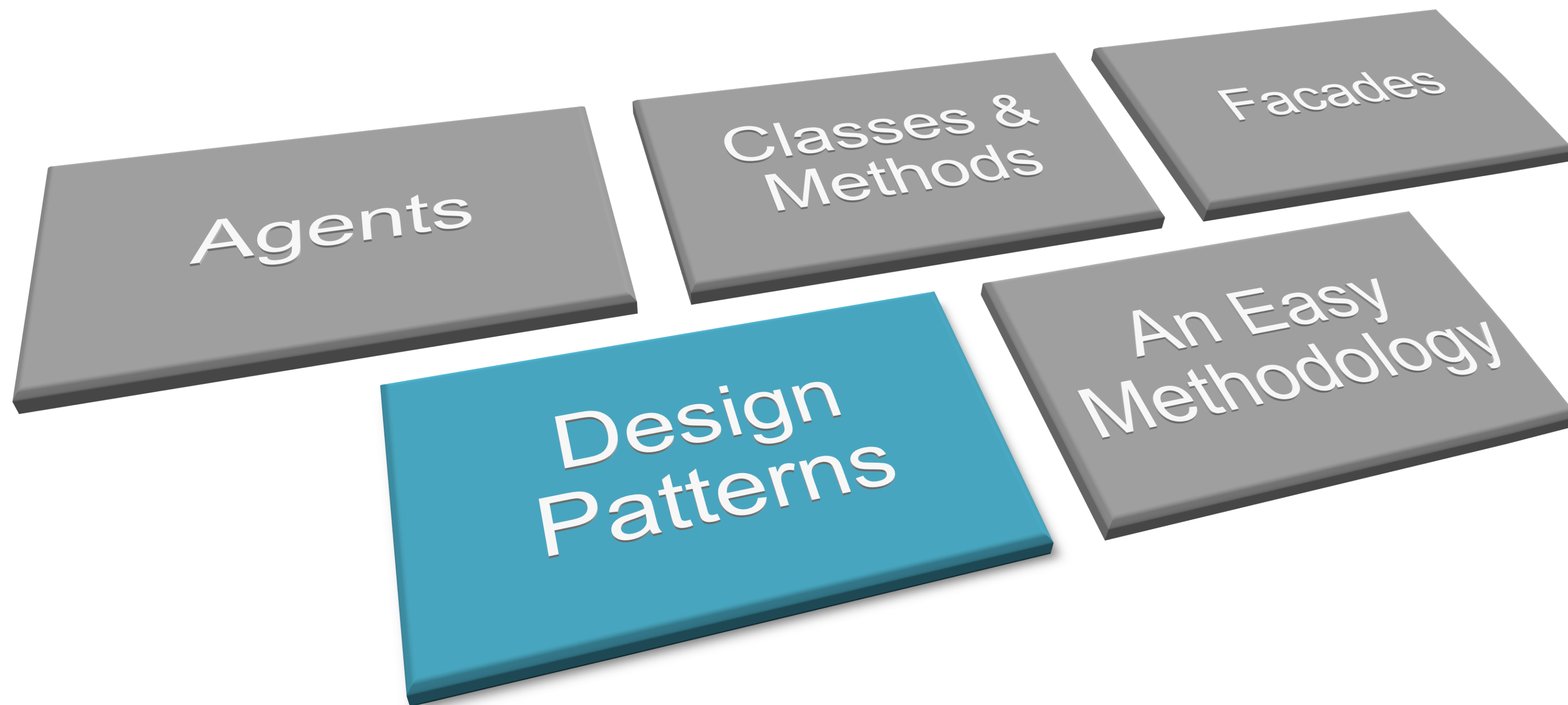
WriteAllocationBackToDocument(VAR ActCostAlloc : Record "Actual Cost Allocation
IF NOT ActCostAllocUI.ConfirmWriteBackAllocation(ActCostAlloc,HideDialog) THEN
    EXIT;

WriteAllocBackMeth.WriteAllocationBackToDocument(ActCostAlloc,TempCostBuffer);
ActCostAllocUI.AcknowledgeWriteBackAllocation(ActCostAlloc,HideDialog);
```

# Facade: UI Separation







# Design Patterns

- A facade is a Design Pattern



Design Patterns have always been the „secret source“ of NAV, enabling developers to learn new areas of the application and be productive without a huge ramp up. This makes design patterns key to understanding the architecture of NAV and writing repeatable code.

PRS is now taking design patterns to a new level by creating awareness of the existing and creating new patterns.

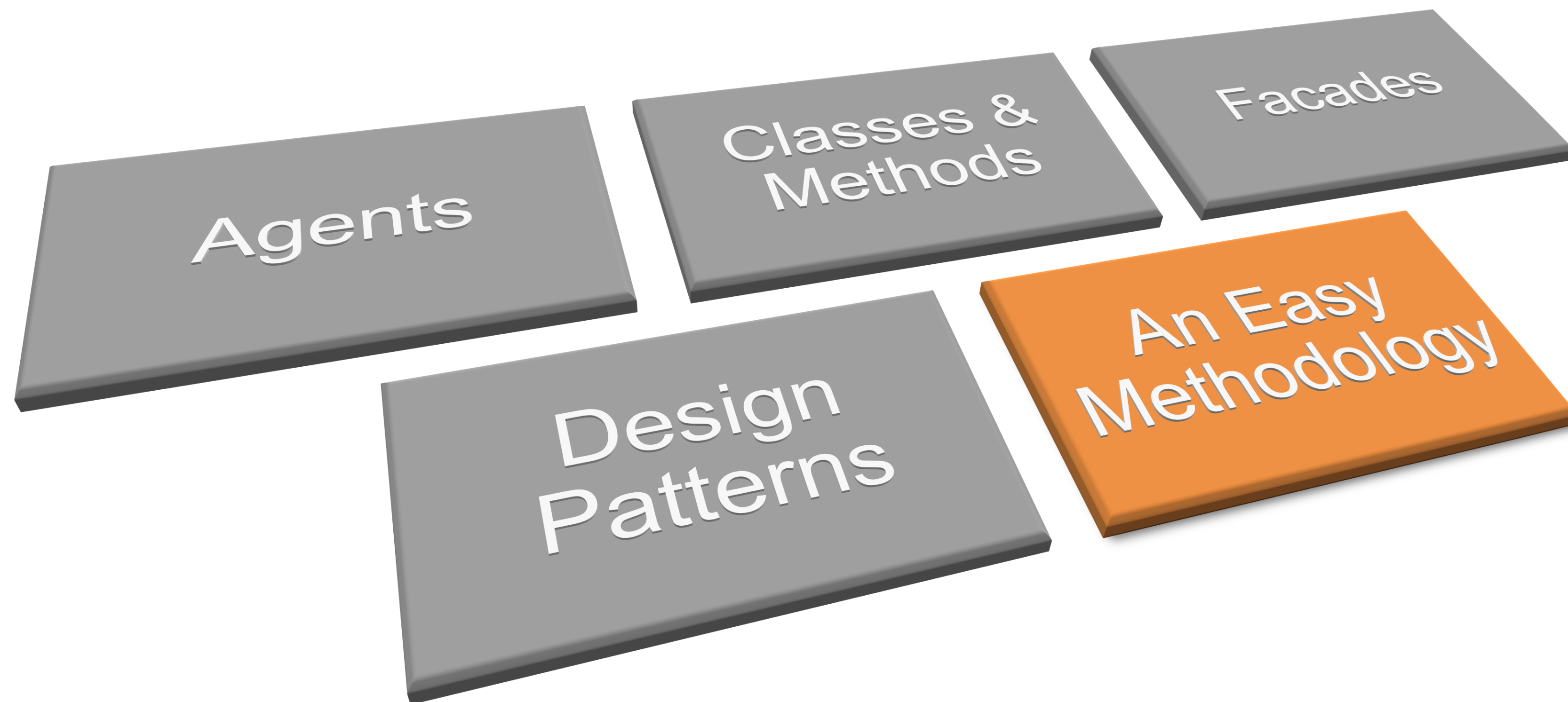


*Michael Nielsen, Director of NAV Development*

# Design Patterns

- PRS Pre-conference day on Repeatable Software and Design Patterns
- Friday, 1:30-3:00 p.m.:  
The Latest Application Code from a Design Patterns Perspective  
Bogdana Botez, Anders Larsen, Mostafa Balat







# An Easy Methodology

- Establish your agents and actions model
- Break your software down into methods and treat them as building blocks
- Think of your method heads as readable flow charts
- Provide decoupling by facades
- Make your terminology consistent from website down to code
- Think and act in terms of design patterns

# Intellectual Property

- Your IP lies in your methods.
- You can give free access to your tables, classes, pages and facades.
- Just protect your methods (if you think you need to).
- Your methods are your IP.

➔ **Full Circle:** Agents, facades and other design patterns are about writing repeatable IP. Applying these concepts allows you to see what your IP really is.

What Is  
Repeatable IP?

How Do I Write  
Repeatable IP?

How Do I Get  
Started?



# How Do I Get Started?

- Do not rewrite NAV standard code
- Start with the code you write tomorrow
- Boy Scout Rule: Always leave an area you touch in better shape than it has been in before
- Set aside a scrum sprint for the key areas of your product



# How Do I Get Started?

- NAV partners are becoming software vendors.
- Everything you know about your industry, needs to be clearly visible in your software. Your chances to teach and explain are disappearing.
- Align your complete organization: If I need more than ten minutes to find in your code what your sales rep is selling, we are both lost.
- Be enthusiastic! – Or as Luc would put it: Be passionate!

# Hooks



# HOOKS RECAP

- Triggered from external code into your IP
- OnPre... OnPost...
- Impacts Upgradability
- Makes PowerShell Merge easier
- Does not solve conflicts, it's not a holy grail
- ...
- So let's talk about an alternative
- ...

# Surrogate Keys





# What is a Surrogate Key?

A **Surrogate key** in a database is a **unique identifier** for either an *entity* in the modeled world or an *object* in the database.

The surrogate key is *not* derived from application data.

# Why use Surrogate Keys?

A concept of adding Generic Functionality such as

- Comments
- Tags
- Documents
- User definable fields
- ...

## How to implement?

Use an Integer field with the AutoIncrement property set to TRUE.

Other options can also be use something such as GUID, but GUID is not great for SQL and all other options including GUID would require code which would increase the footprint.

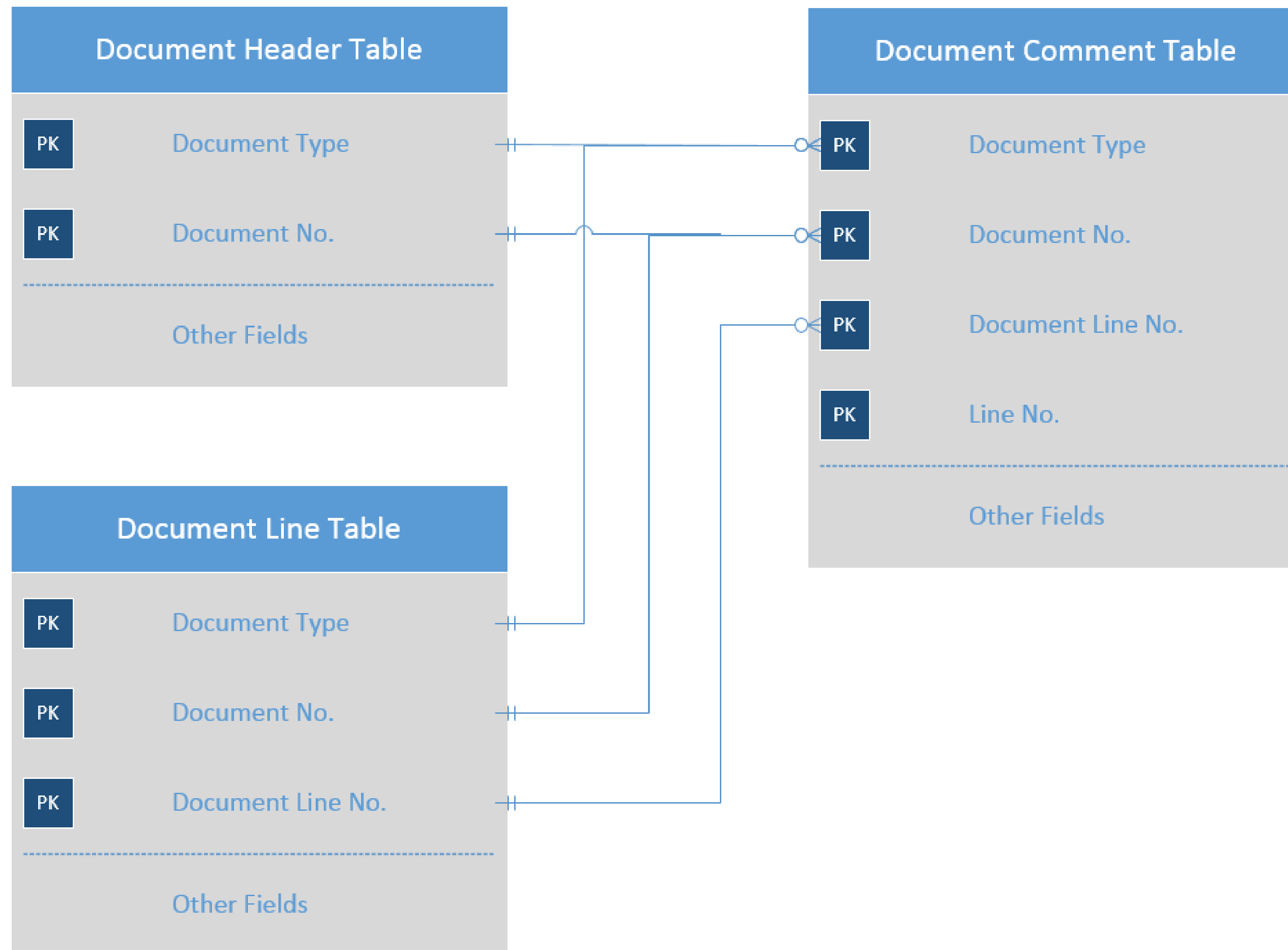
# Surrogate Key vs. RecordID

	Surrogate Key	RecordID
How to add	Must be added to each table as an Integer Field with AutoIncrement set to TRUE	Already exist
Rename	No Impact	Must be managed on the Rename trigger via Code or globally trigger in Codeunit 1
Delete	Managed generically under the Global Delete trigger in Codeunit 1	Typically managed under each table delete trigger but can also be managed globally in Codeunit 1
Filtering	Filter can be set on the fly. No Code. Small footprint.	Filter require Code

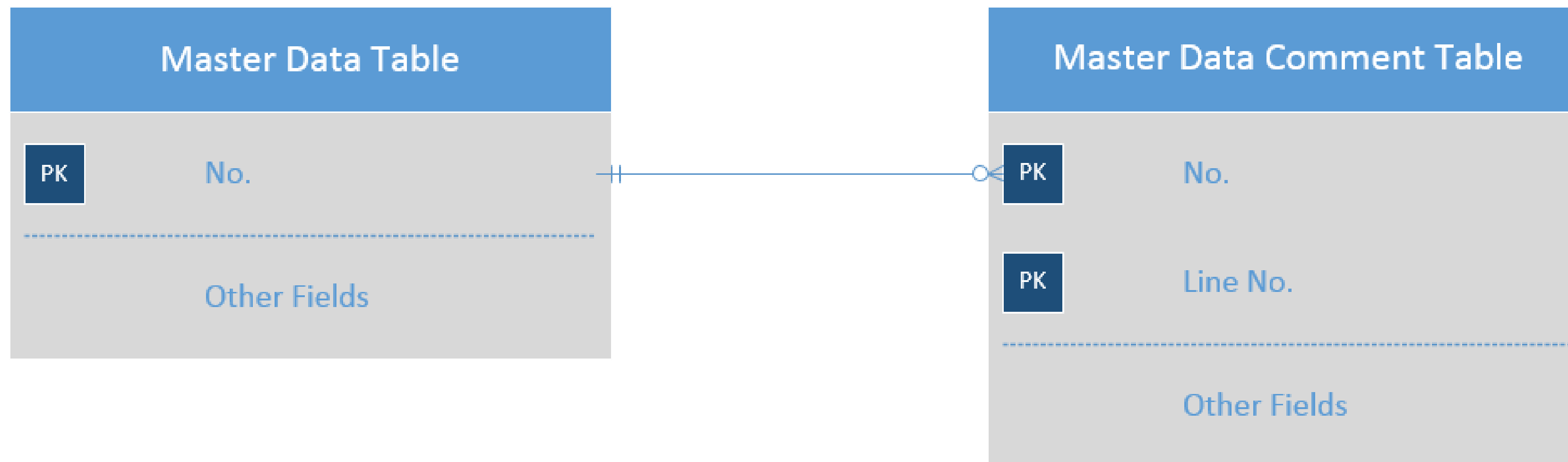


# Example

# *Table structure for linking a Document Header and Line Table with a Document Comment Table.*



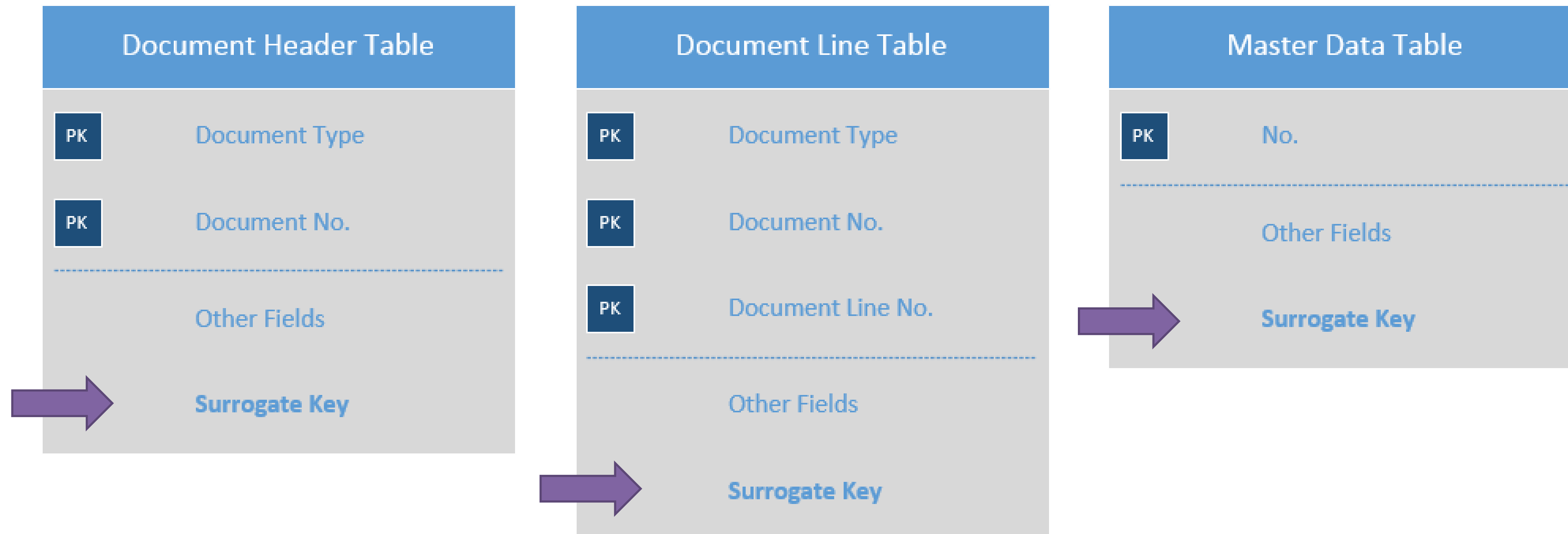
# *Table Structure for linking a Master Data Table with a Master Data Comment Table*



# How to do this with a Surrogate Key!



# Step 1: Adding the Surrogate Key



## Step 2: Create a generic link-able sub table

Comment Table	
PK	Table ID
PK	Surrogate Key Reference
PK	Line No.
Other Fields	

	E..	Field No.	Field Name	Data Type	Length	Description
	✓	1	Table ID	Integer		
▶	✓	2	SurrogateKeyRef	Integer		
	✓	4	Line No.	Integer		
	✓	5	Date	Date		
	✓	6	Code	Code	10	
	✓	7	Comment	Text	80	

## Step 3: Create a List and Sheet Comment page showing the table information.

[illegible]

This will look exactly like a standard Comment List and Sheet today!

## Step 4: Add an Action

This could also be an Factbox or SubPage.

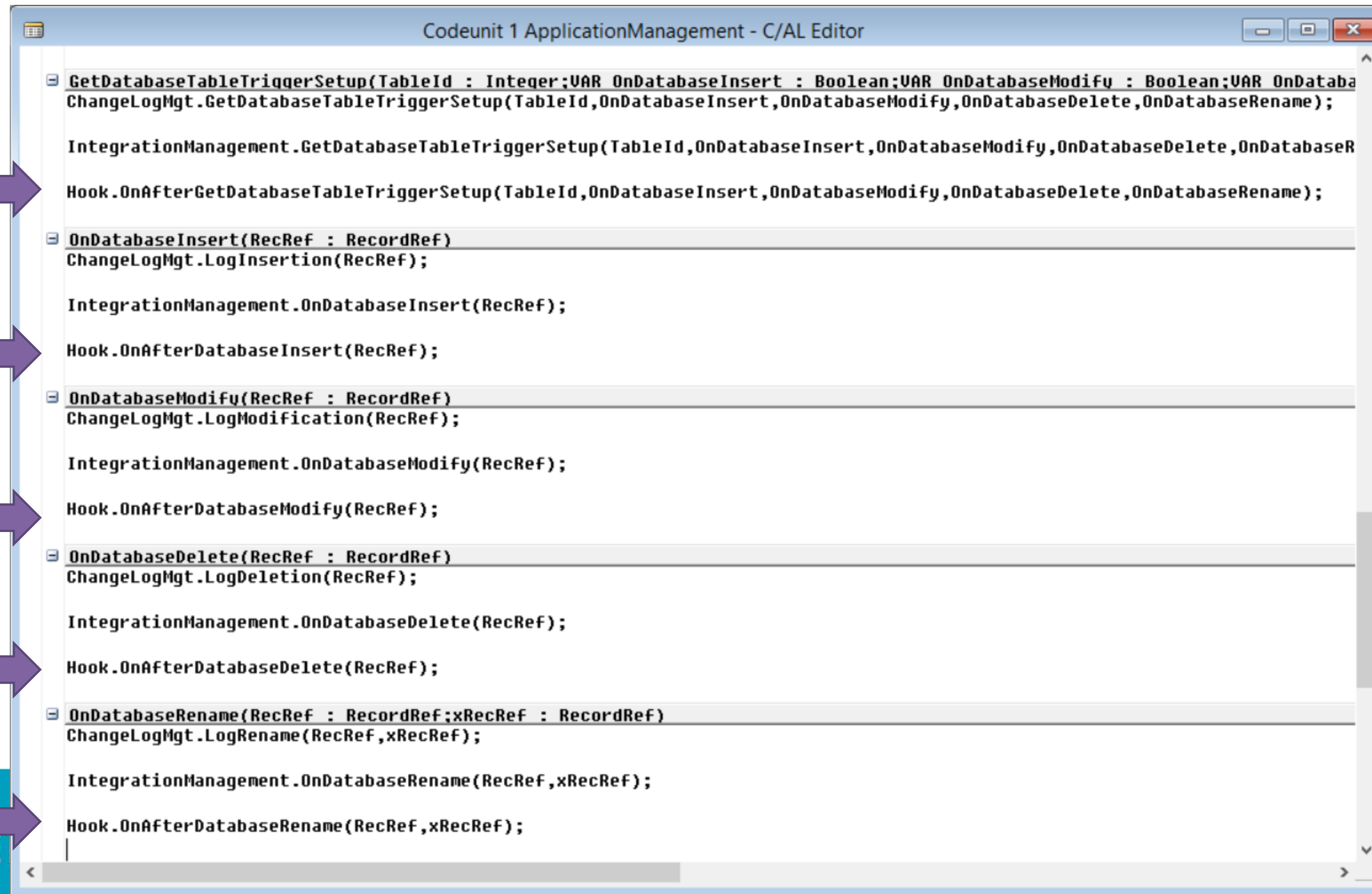
The screenshot displays two overlapping windows from the SAP Page Designer tool. The background window, titled 'Page - Action Designer', shows a table with columns 'E.. Type', 'SubType', 'SourceExpr', 'Name', and 'Caption'. It lists 'ActionContainer' (SubType: RelatedInformation, Name: <Action12>, Caption: <Action12>), 'ActionGroup' (Name: <Action11>, Caption: Master Data), and an 'Action' (Name: Comments, Caption: Comments). The foreground window, titled 'Comments - Properties', shows a table with columns 'Property' and 'Value'. It lists various properties for the 'Comments' action, including 'Promoted' (<No>), 'PromotedCategory' (<New>), 'PromotedIsBig' (<No>), 'Scope' (<Page>), 'Ellipsis' (<No>), 'ShortCutKey' (<>), 'RunObject' (Page Generic Comment Sheet), 'RunPageView' (SORTING(Table ID,SurrogateKeyRef,Line No.) ORDER(Ascending) WHERE(Table ID=CONST(23036101))), 'RunPageLink' (SurrogateKeyRef=FIELD(SurrogateKey)), and 'RunPageOnRec' (<No>).

E.. Type	SubType	SourceExpr	Name	Caption
ActionContainer	RelatedInformation		<Action12>	<Action12>
ActionGroup			<Action11>	Master Data
Action			Comments	Comments

Property	Value
Promoted	<No>
PromotedCategory	<New>
PromotedIsBig	<No>
Scope	<Page>
Ellipsis	<No>
ShortCutKey	<>
RunObject	Page Generic Comment Sheet
RunPageView	SORTING(Table ID,SurrogateKeyRef,Line No.) ORDER(Ascending) WHERE(Table ID=CONST(23036101))
RunPageLink	SurrogateKeyRef=FIELD(SurrogateKey)
RunPageOnRec	<No>



# Step 5: Create a Hook for Functions in Codeunit 1 ApplicationManagement if one doesn't already exist



```
Codeunit 1 ApplicationManagement - C/AL Editor

GetDatabaseTableTriggerSetup(TableId : Integer;VAR OnDatabaseInsert : Boolean;VAR OnDatabaseModify : Boolean;VAR OnDatabaseDelete : Boolean;VAR OnDatabaseRename : Boolean)
ChangeLogMgt.GetDatabaseTableTriggerSetup(TableId,OnDatabaseInsert,OnDatabaseModify,OnDatabaseDelete,OnDatabaseRename);

IntegrationManagement.GetDatabaseTableTriggerSetup(TableId,OnDatabaseInsert,OnDatabaseModify,OnDatabaseDelete,OnDatabaseRename);

Hook.OnAfterGetDatabaseTableTriggerSetup(TableId,OnDatabaseInsert,OnDatabaseModify,OnDatabaseDelete,OnDatabaseRename);

OnDatabaseInsert(RecRef : RecordRef)
ChangeLogMgt.LogInsertion(RecRef);

IntegrationManagement.OnDatabaseInsert(RecRef);

Hook.OnAfterDatabaseInsert(RecRef);

OnDatabaseModify(RecRef : RecordRef)
ChangeLogMgt.LogModification(RecRef);

IntegrationManagement.OnDatabaseModify(RecRef);

Hook.OnAfterDatabaseModify(RecRef);

OnDatabaseDelete(RecRef : RecordRef)
ChangeLogMgt.LogDeletion(RecRef);

IntegrationManagement.OnDatabaseDelete(RecRef);

Hook.OnAfterDatabaseDelete(RecRef);

OnDatabaseRename(RecRef : RecordRef;xRecRef : RecordRef)
ChangeLogMgt.LogRename(RecRef,xRecRef);

IntegrationManagement.OnDatabaseRename(RecRef,xRecRef);

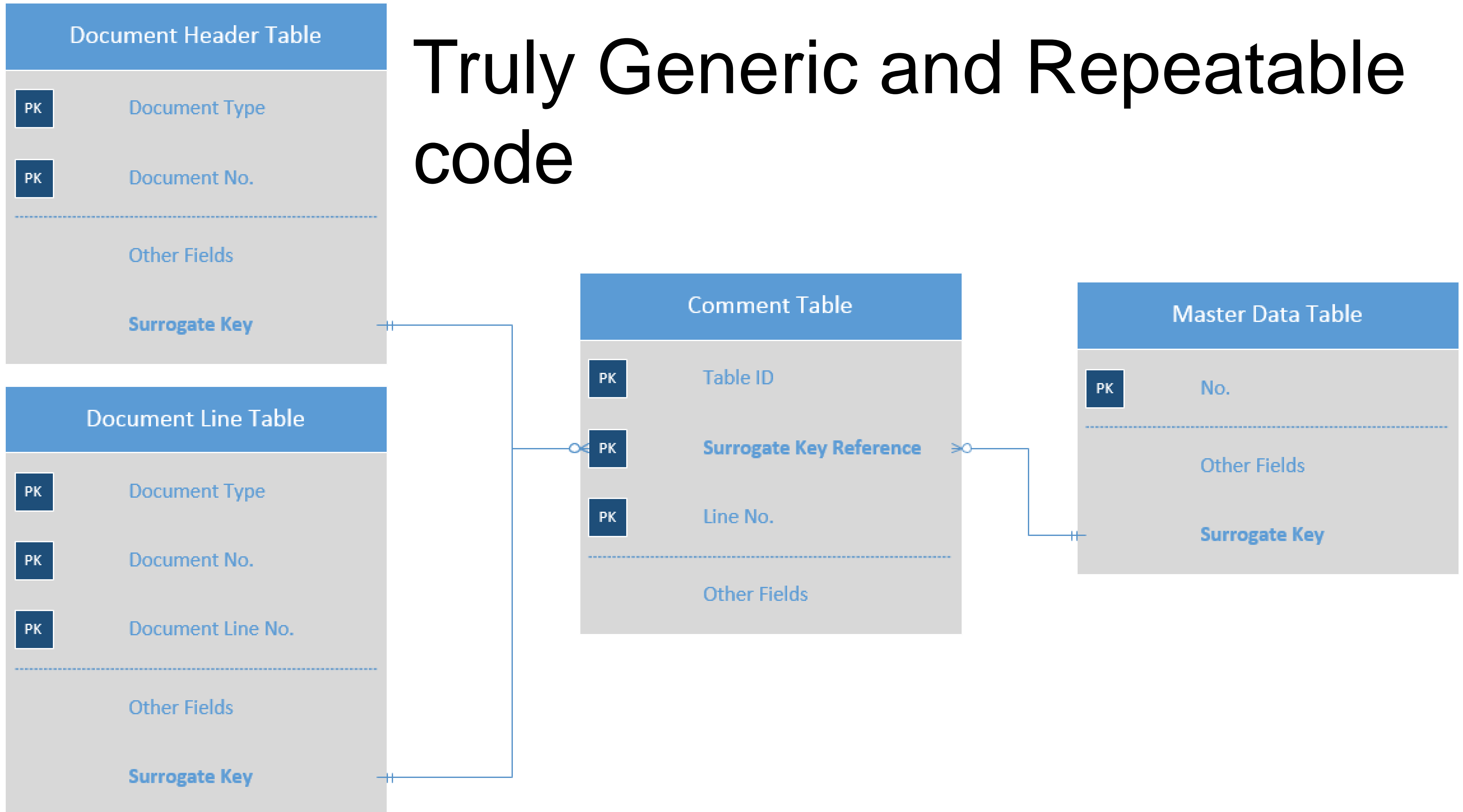
Hook.OnAfterDatabaseRename(RecRef,xRecRef);
```

WHEN YOU ARE

**Step 6: Create a code to delete records in the Sub table if a main table record is deleted if this is required.**

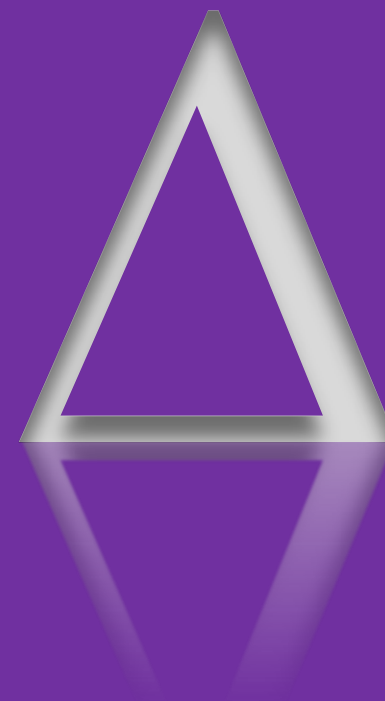
This function needs to be called from the Hook created in Step 5.

# Truly Generic and Repeatable code









Shipping Delta's



## Detailed example - Deltas

Table 888  
Fields  
PhoneNo  
FirstName  
LastName

Table 888  
Fields  
PhoneNo  
Name

Table 888  
Add Field  
E-Mail



Table 888  
Fields  
PhoneNo  
FirstName  
LastName  
E-mail

Table 888  
Fields  
PhoneNo  
Name  
E-Mail



## Detailed example - Deltas

**ORIGINAL**

Fields  
PhoneNo  
FirstName  
LastName

**TARGET**

Fields  
PhoneNo  
Name

Compute  
Delta

**DELTA**

Add Field  
E-Mail

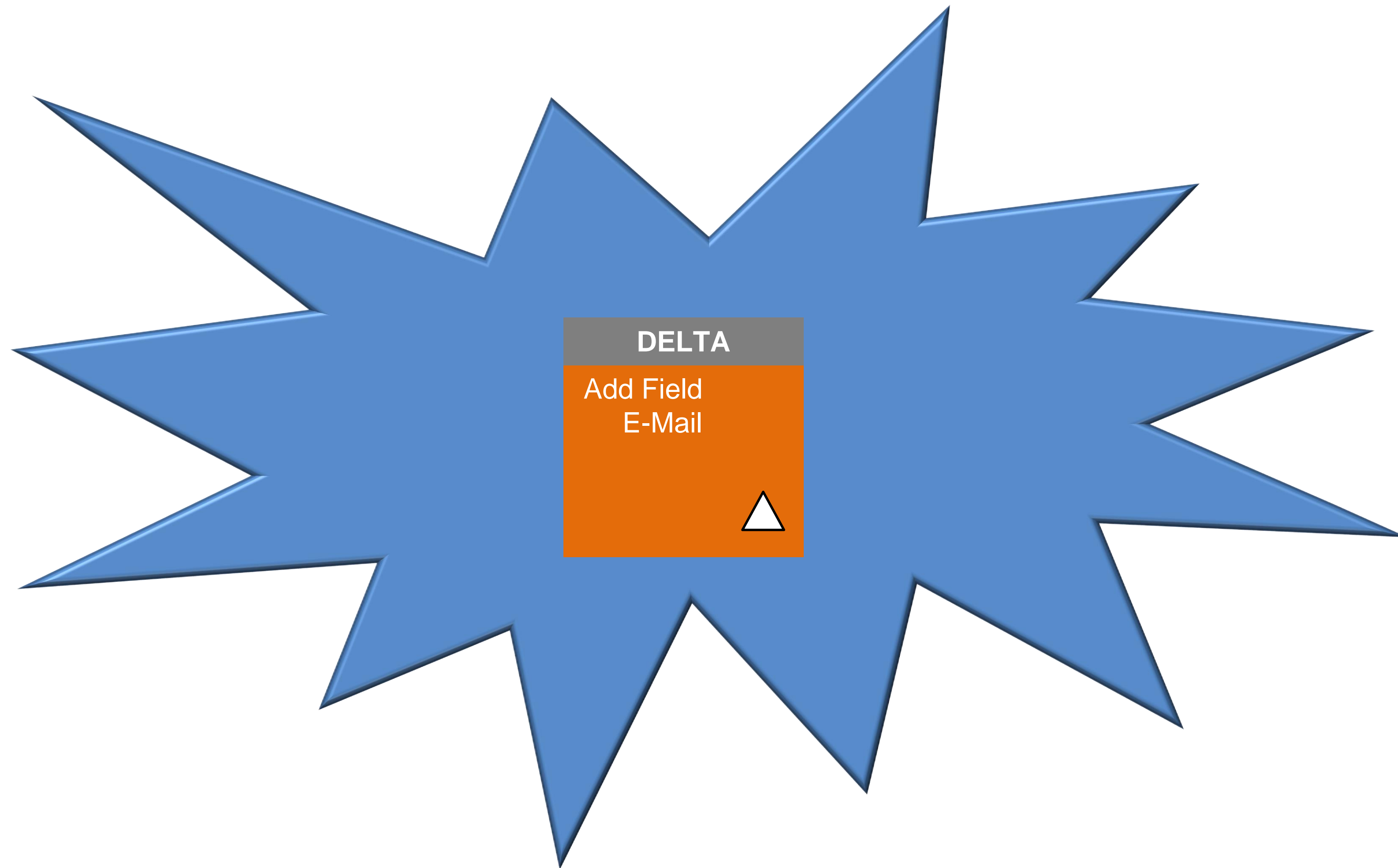
Apply  
Delta

**MODIFIED**

Fields  
PhoneNo  
FirstName  
LastName  
E-mail

**RESULT**

Fields  
PhoneNo  
Name  
E-Mail





# SHIPPING SOFTWARE VIA DELTA

## GOOD

- Save Time
- Applies to all Databases

## BAD

- Version List
- Conflicts are hard to handle

## UGLY

- Tables and Fields require Partner License

# MORE TOMORROW...

Eric Wauters

BELGIUM

iFacto Business  
Solutions



09:00 - 10:30

Looking at NAV the Powershell way ...

LEVEL: 200 - INTERMEDIATE

LEVEL: 300 - ADVANCED

ROOM 9



# Data Conversion

# Upgrade Codeunit

Property	Value
ID	3
Local	<No>
FunctionType	TableSyncSetup

Property	Value
ID	104025
Name	UPG7180.W1
Permissions	<Undefined>
CFRONTMayUsePermissions	<No>
TableNo	<Undefined>
SingleInstance	<No>
Subtype	Upgrade

```
Codeunit 104025 UPG7180.W1 - C/AL Editor

OnRun()
[TableSyncSetup] GetTableSyncSetup(UAR TableSynchSetup : Record "Table Synch. Setup")

    DataUpgradeMgt.SetTableSyncSetup(DATABASE::"Payment Terms",170001,TableSynchSetup.Mode::Copy);
    DataUpgradeMgt.SetTableSyncSetup(DATABASE::"Vendor",170002,TableSynchSetup.Mode::Copy);
    DataUpgradeMgt.SetTableSyncSetup(DATABASE::"Vendor Invoice Disc.",170003,TableSynchSetup.Mode::Copy);
    DataUpgradeMgt.SetTableSyncSetup(DATABASE::"Job",170004,TableSynchSetup.Mode::Copy);
    DataUpgradeMgt.SetTableSyncSetup(455,0,TableSynchSetup.Mode::Force);

[CheckPrecondition] CheckUPGPreconditions()
{...}

[Upgrade] UpgradePaymentTerms()
IF UPGPaymentTerms.FINDSET THEN
    REPEAT
        {...}
    UNTIL UPGPaymentTerms.NEXT = 0;

[Upgrade] UpgradeVendors()
IF UPGVendors.FINDSET THEN
    REPEAT
        {...}
    UNTIL UPGVendors.NEXT = 0;

[Upgrade] UpgradeVendorInvDisc()
IF UPGVendorInvoiceDisc.FINDSET THEN
    REPEAT
        {...}
    UNTIL UPGVendorInvoiceDisc.NEXT = 0;

[Upgrade] UpgradeJobs()
IF UPGJobs.FINDSET THEN
    REPEAT
        {...}
    UNTIL UPGJobs.NEXT = 0;
```

Property	Value
ID	7
Local	<No>
FunctionType	CheckPrecondition

Property	Value
ID	1
Local	<No>
FunctionType	Upgrade



# UPGRADE PROCESS

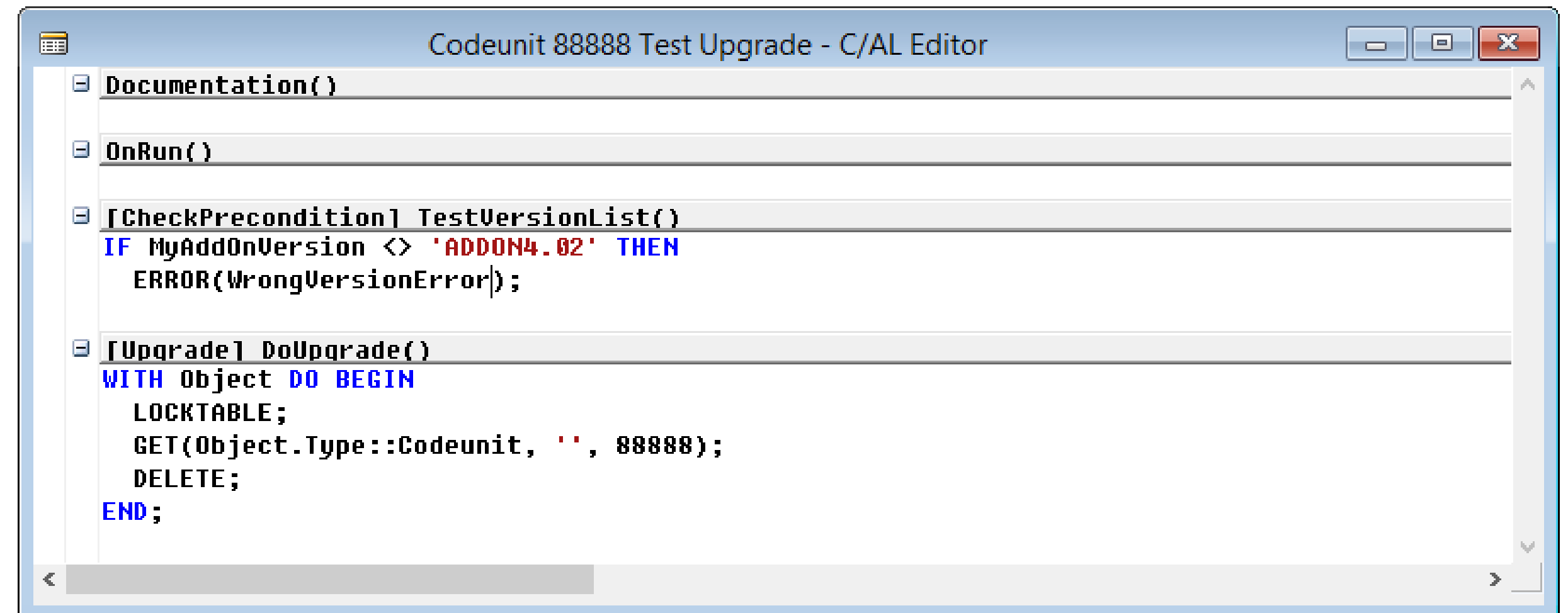


# DATA CONVERSION WITH ANY FOB

ANY RELEASE

TEST VERSION

SELF DESTRUCT



```
Codeunit 88888 Test Upgrade - C/AL Editor

Documentation()
OnRun()

[CheckPrecondition] TestVersionList()
IF MyAddOnVersion <> 'ADDON4.02' THEN
    ERROR(WrongVersionError);

[Upgrade] DoUpgrade()
WITH Object DO BEGIN
    LOCKTABLE;
    GET(Object.Type::Codeunit, '', 88888);
    DELETE;
END;
```



Design patterns through .NET

# DESIGN PATTERNS THROUGH .NET

True or False: To have true design patterns, you need an object-oriented language?

.NET and C# are flexible, allow for a vast number of language-level design patterns

C/AL is very inflexible, not object oriented, and only allows for a few basic patterns (hooks, facade)

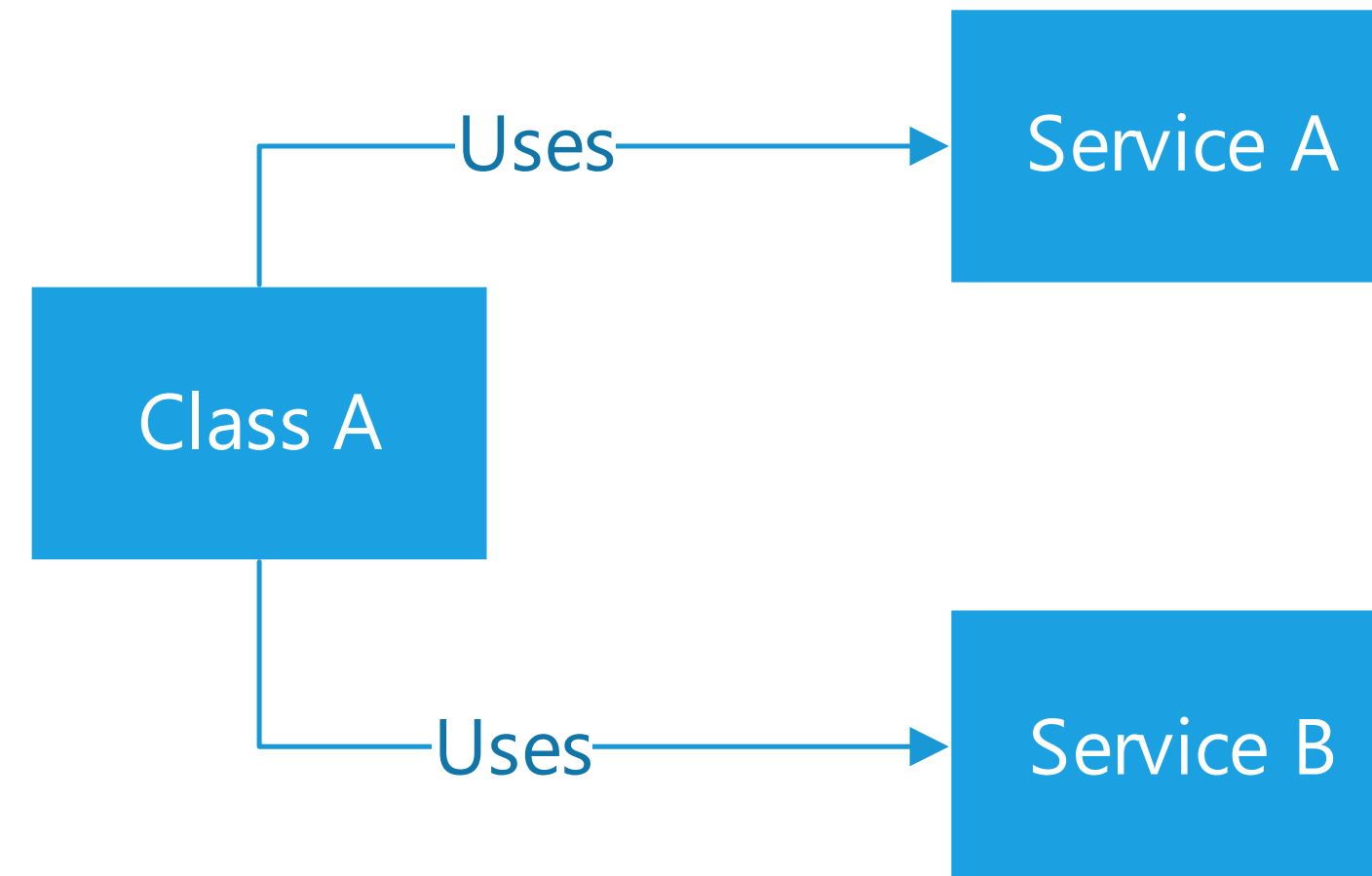
DotNet interoperability opens a door to a wide range of .NET patterns to be applied in C/AL



# SERVICE LOCATOR PATTERN

- Decouples classes from dependencies
- Allows you to write logic that depends on classes whose concrete implementation is not known at compile time
- Allows you to test classes in isolation, without the dependencies
- Divides the application into loosely coupled modules that can be independently developed, tested, versioned, and deployed

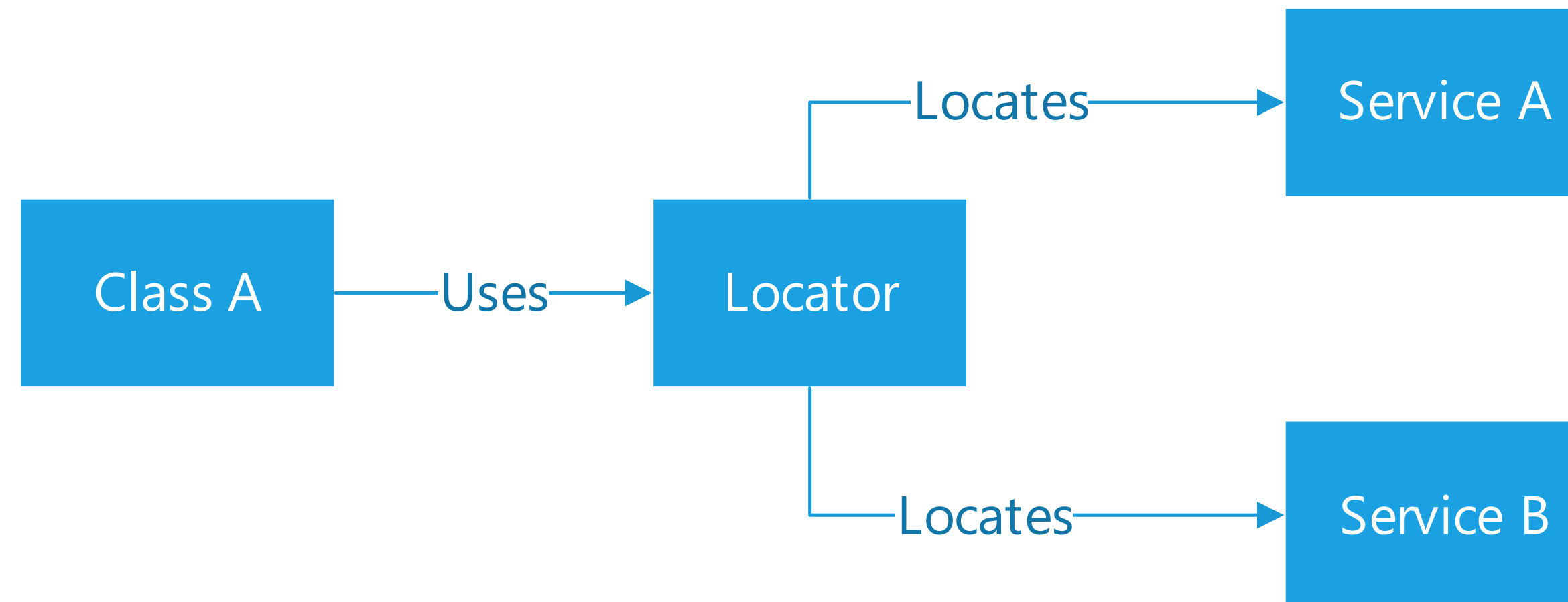
# THE SITUATION



# DEMO

A tightly coupled dependency in a real-life scenario

# THE SOLUTION





# DEMO

A decoupled dependency through the Service Locator design pattern in a real-life scenario

# Roadmap



# WHAT WE HAVE TODAY

Expand & Collapse Functions

“Automatic” Merging

Data Conversion for FOB

Longer Function Names

Shipping Delta Files

Refactored Application Parts

Functions are Local

Documented Patterns

Out of the Box UI Thinking

Automatic Population of Variable Names

Fresh Patterns

Mandatory Fields

# VISION FOR TOMORROW

Continuous Documentation of  
Design Patterns

Connect to TFS

Enumerators

Continuous Refactoring

Data Dictionary

Easier integration

Reduce Code Cloning

Try/Catch/Finally

Add Tables and Fields without  
License Restrictions

Adding Design Pattern Metadata

ForEach





Team Foundation Server

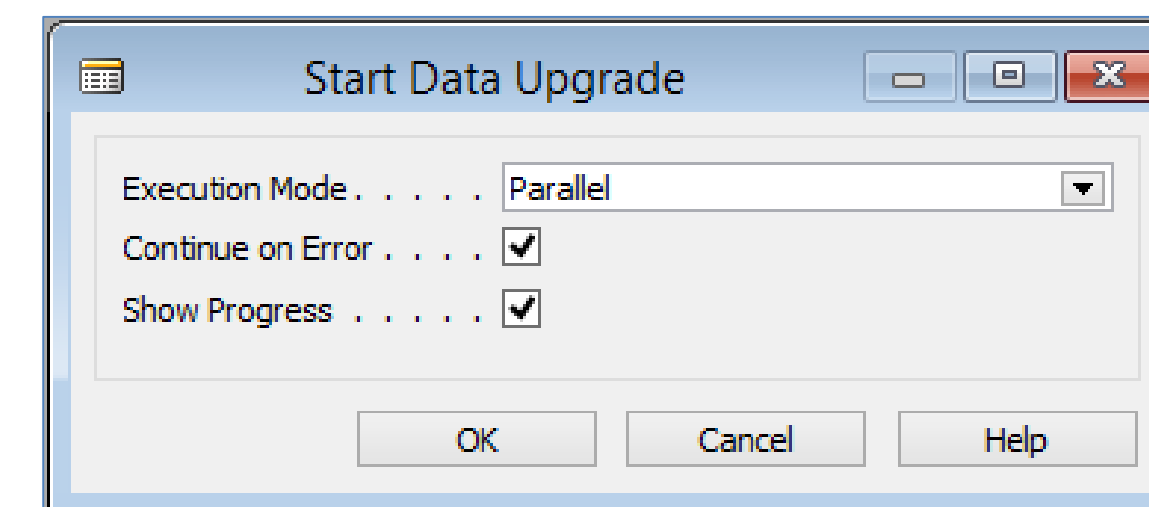
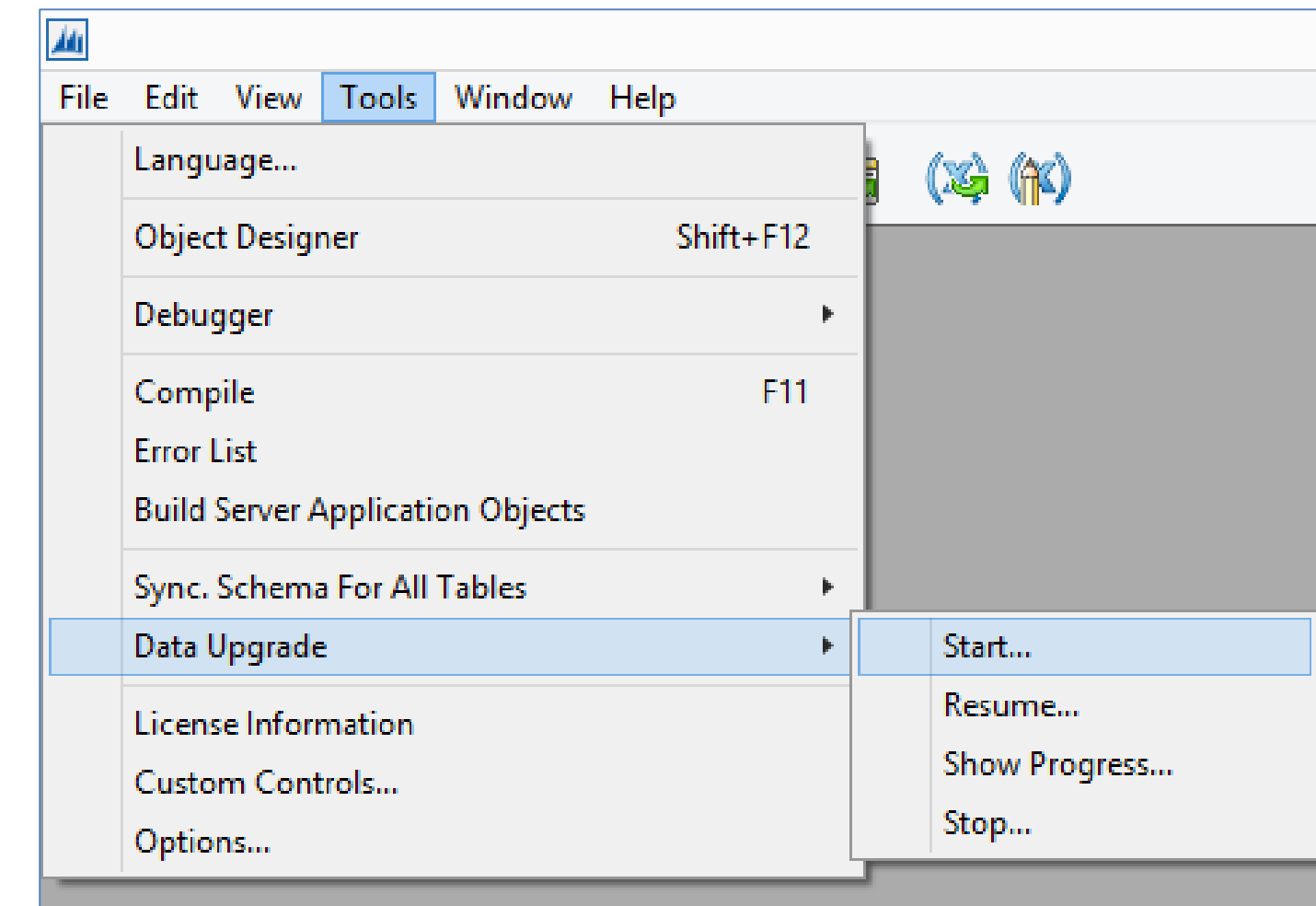


**Start-NAVDataUpgrade**

**Get-NAVDataUpgrade**

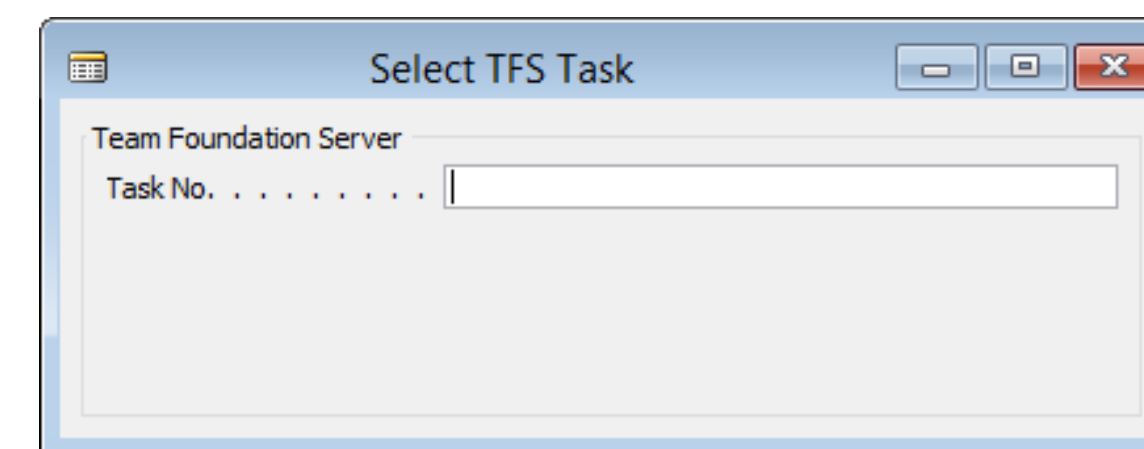
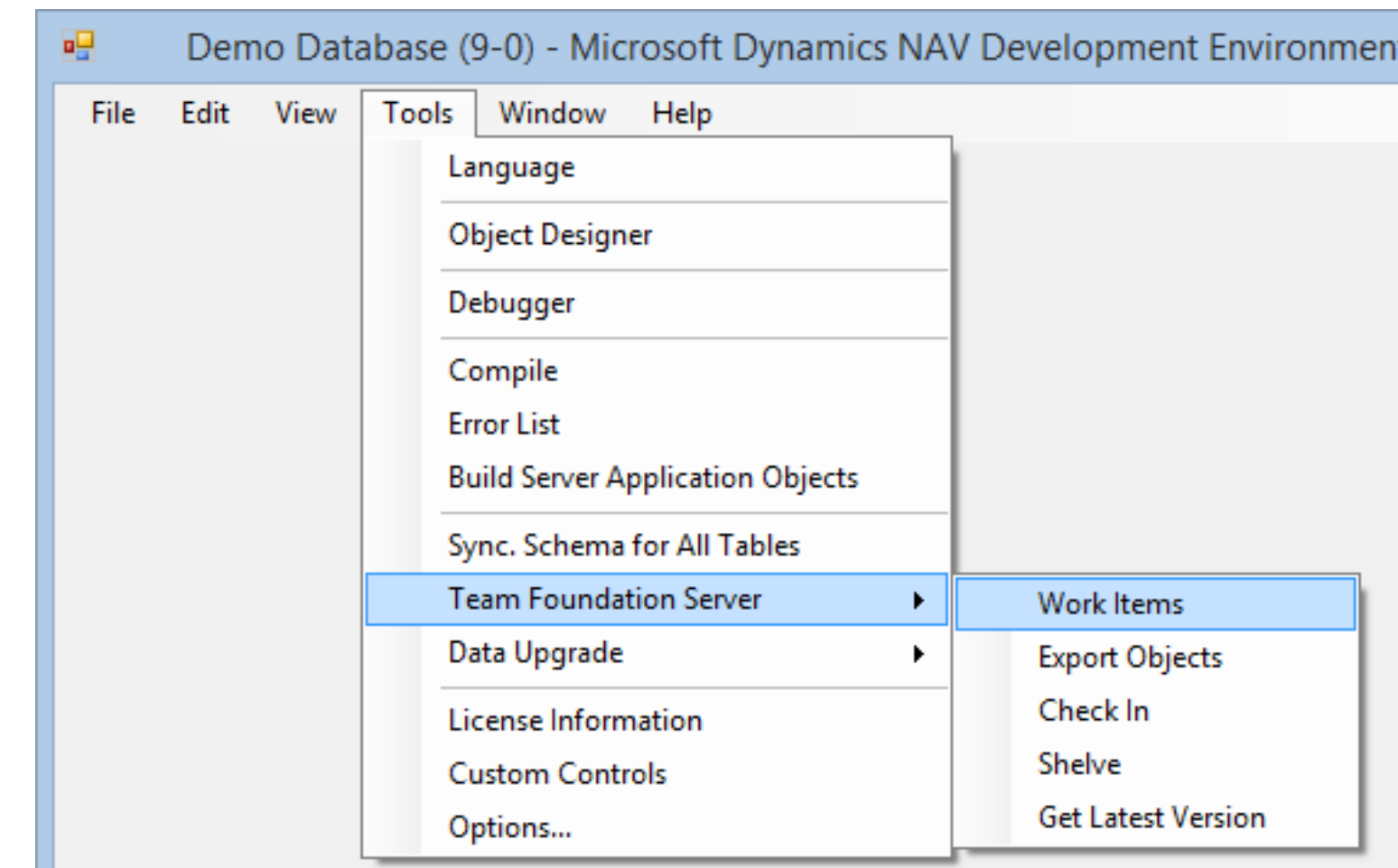
**Resume-  
NAVDataUpgrade**

**Stop-NAVDataUpgrade**





**Export-NAVObject**  
**CheckIn-NAVObject**  
**Connect-TFSProject**  
**Activate-TFSTask**  
**Get-TFSVersion**



# CLOUD READY DEVELOPMENT METHODOLOGY

## WRITING REPEATABLE SOFTWARE

Day 1 – Design Patterns for Repeatable Software

Germany

Day 2 – RapidStart Code – PowerShell merging

The Netherlands

Day 3 – RapidStart Upgrade – Automated Upgrades

Belgium

Day 4/5 Optional work on partners add-on.

India - New Delhi

Japan - Tokyo

USA - Atlanta

Canada - Toronto



“A Professional Partner Channel  
Deserves a Professional  
Work Environment”



**Partner Ready Software**

It's YOU!



**Partner Ready Software**

# Any Questions?

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)

NAV  
TECH  
DAYS  
2014  
[mibuso.com](http://mibuso.com)



# THANK YOU

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)

NAV  
TECH  
DAYS  
2014  
[mibuso.com](http://mibuso.com)





# COFFEE BREAK

see you back in 30 min.

WHEN YOU ARE PASSIONATE ABOUT MICROSOFT DYNAMICS NAV | [www.navtechdays.com](http://www.navtechdays.com)

NAV  
TECH  
DAYS  
2014

[mibuso.com](http://mibuso.com)