HOW TO WRITE REPEATABLE SOFTWARE

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

INTRODUCTION

Vjekoslav Babic
CROATIA
Fortempo - Business Consulting

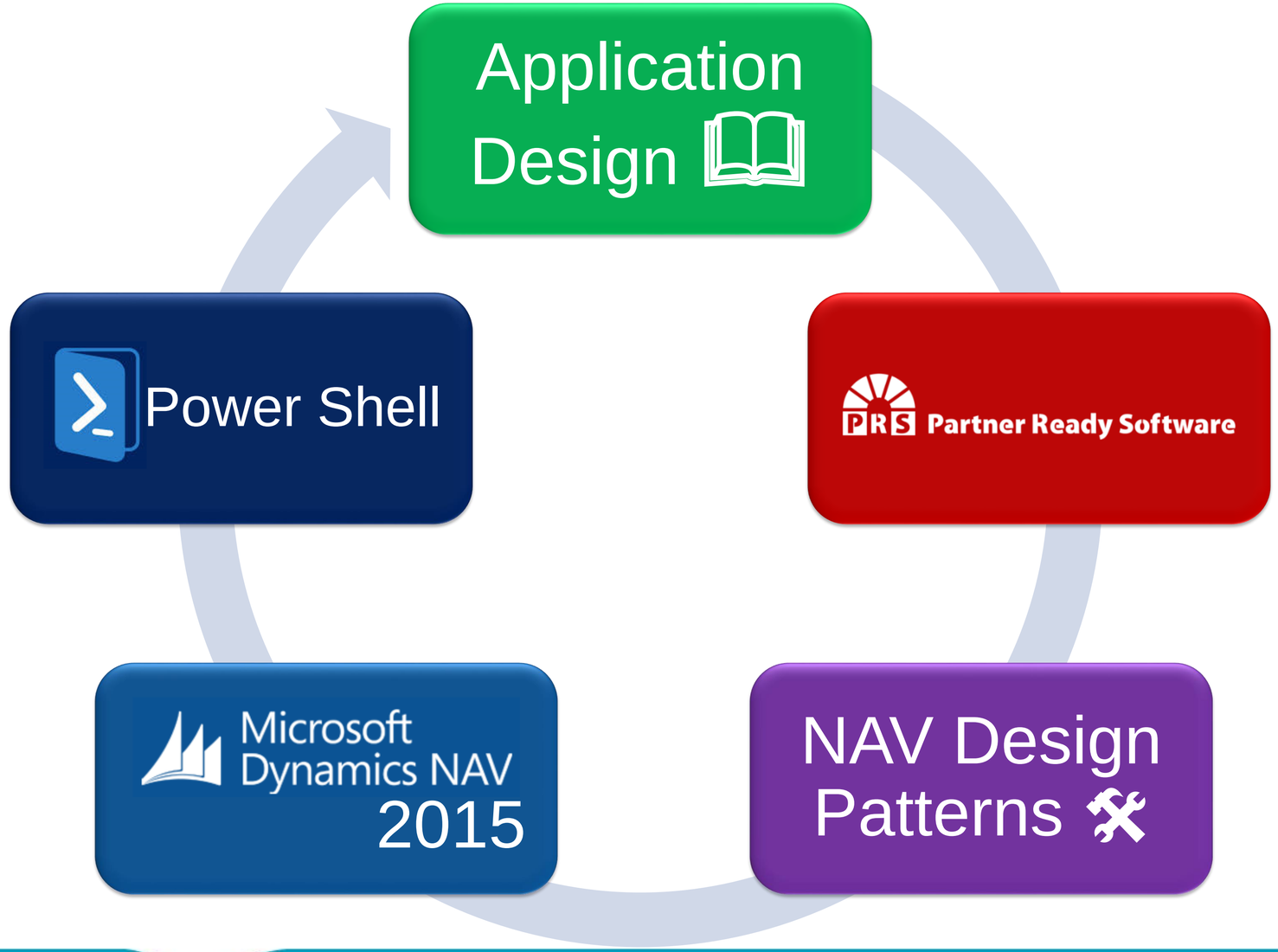
Mark Brummel
NETHERLANDS
Brummel Automatisering

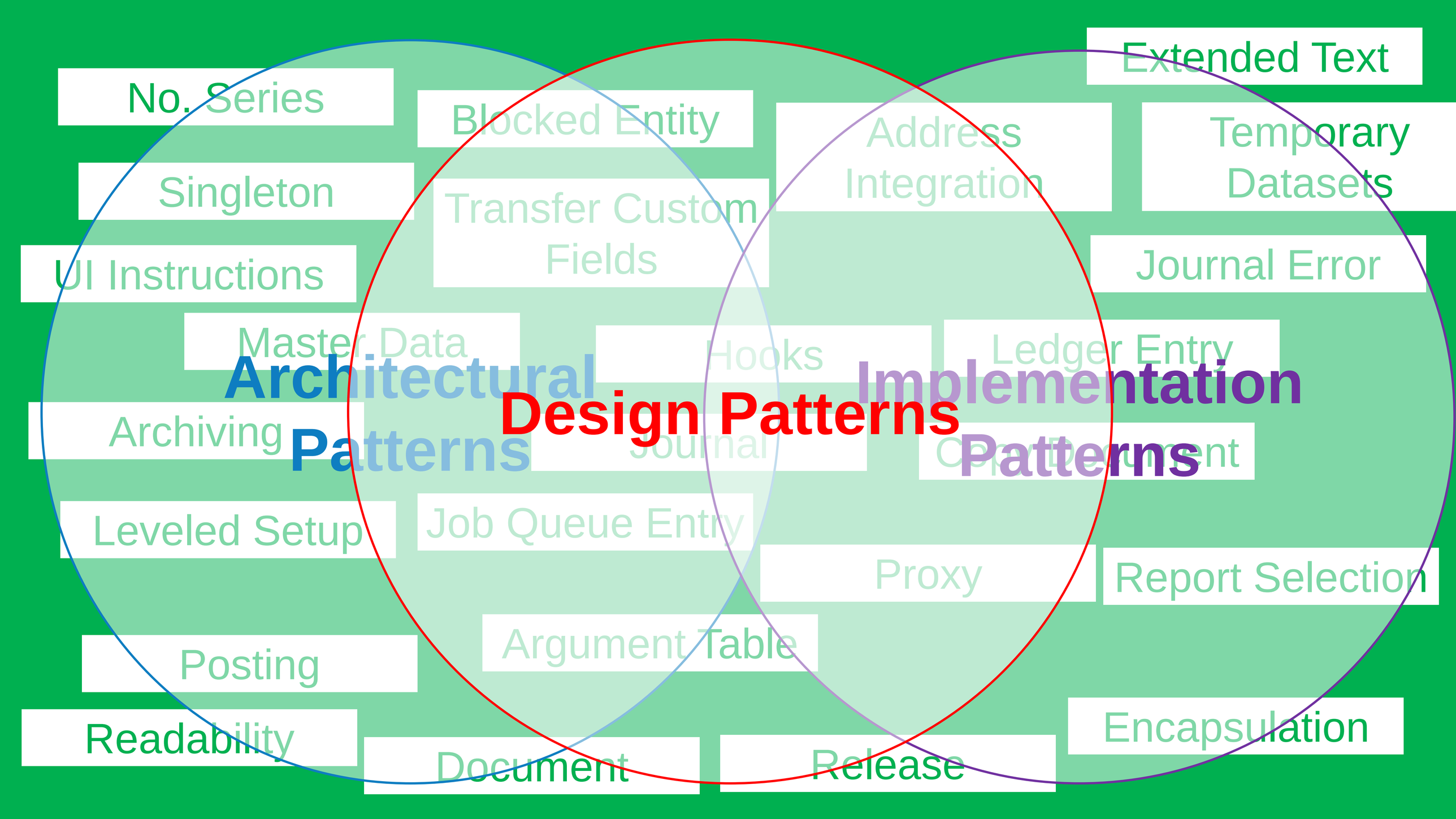
Soren Klemmensen
CANADA
Concept Computer Corp

Gary Winter
GERMANY
agiles Group

AGENDA

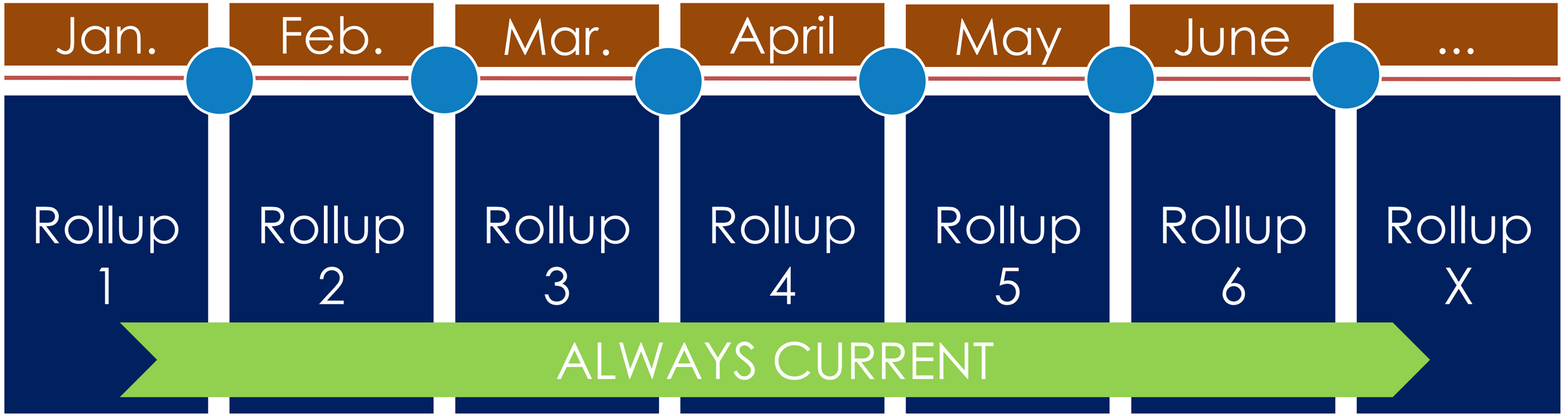
- Overview
- Methodology
- Out of the box
- Roadmap


Application Design
Partner Ready Software
NAV Design Patterns
Microsoft Dynamics NAV 2015
Power Shell


Extended Text
No. Series
Blocked Entity
Address Integration
Temporary Datasets
Singleton
Transfer Custom Fields
Journal Error
UI Instructions
Master Data
Hooks
Ledger Entry
Architectural Patterns
Design Patterns
Implementation Patterns
Archiving
Journal
Leveled Setup
Job Queue Entry
Proxy
Report Selection
Argument Table
Posting
Readability
Encapsulation
Document
Release

MICROSOFT DYNAMICS NAV VERSIONS

MICROSOFT DYNAMICS NAV ROLLUPS



HOW TO STAY CURRENT

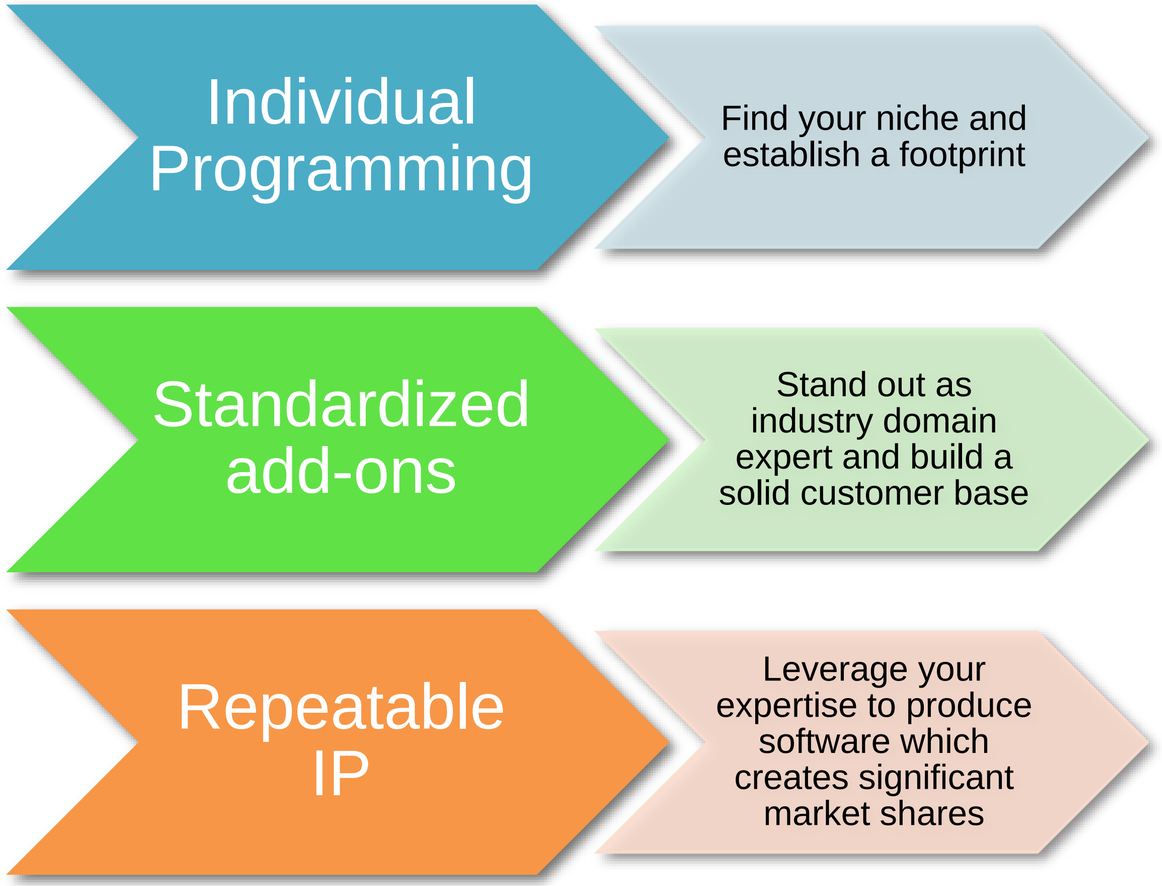
Upgrade in Minutes

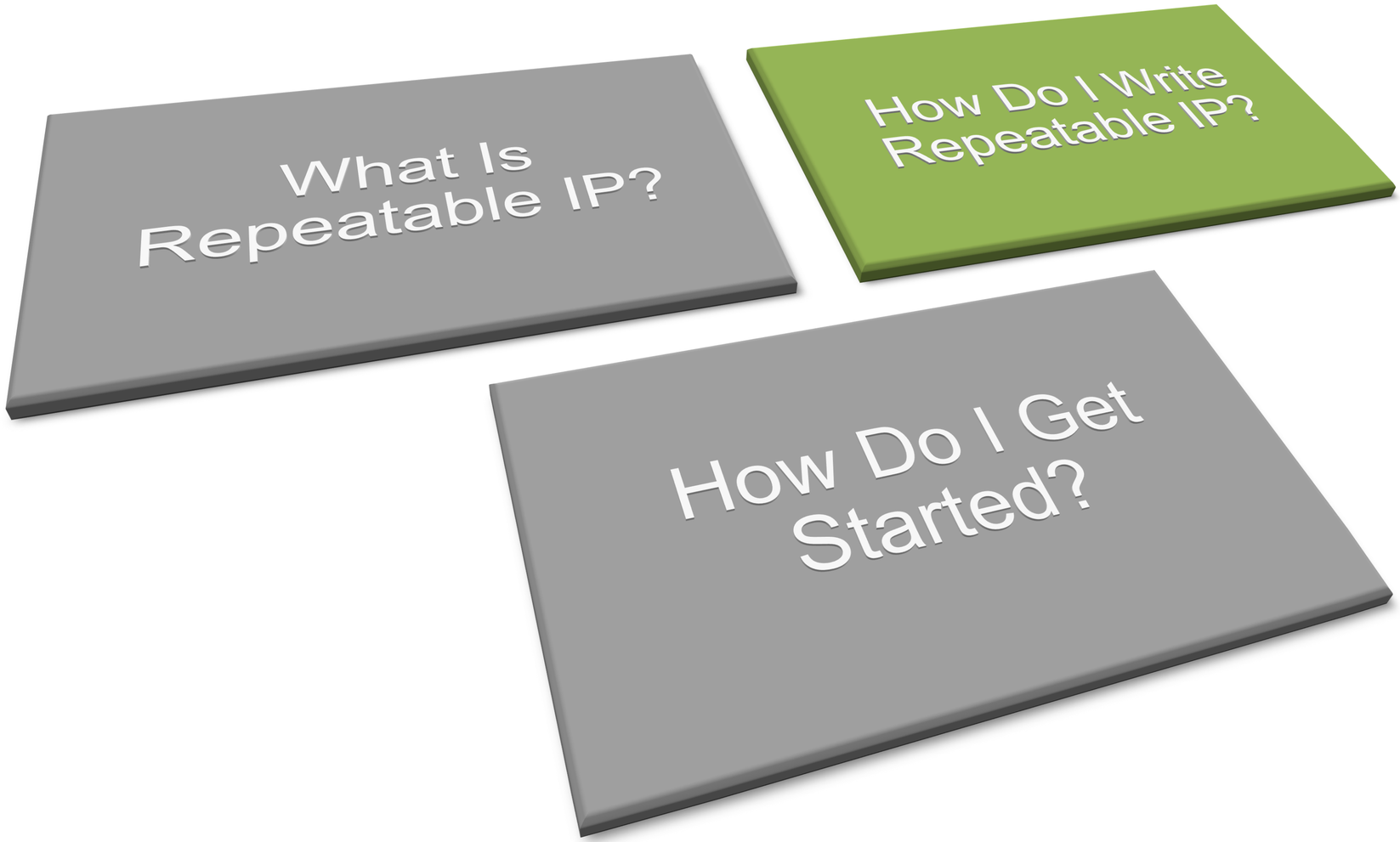
Total Cost of Ownership

Methodology

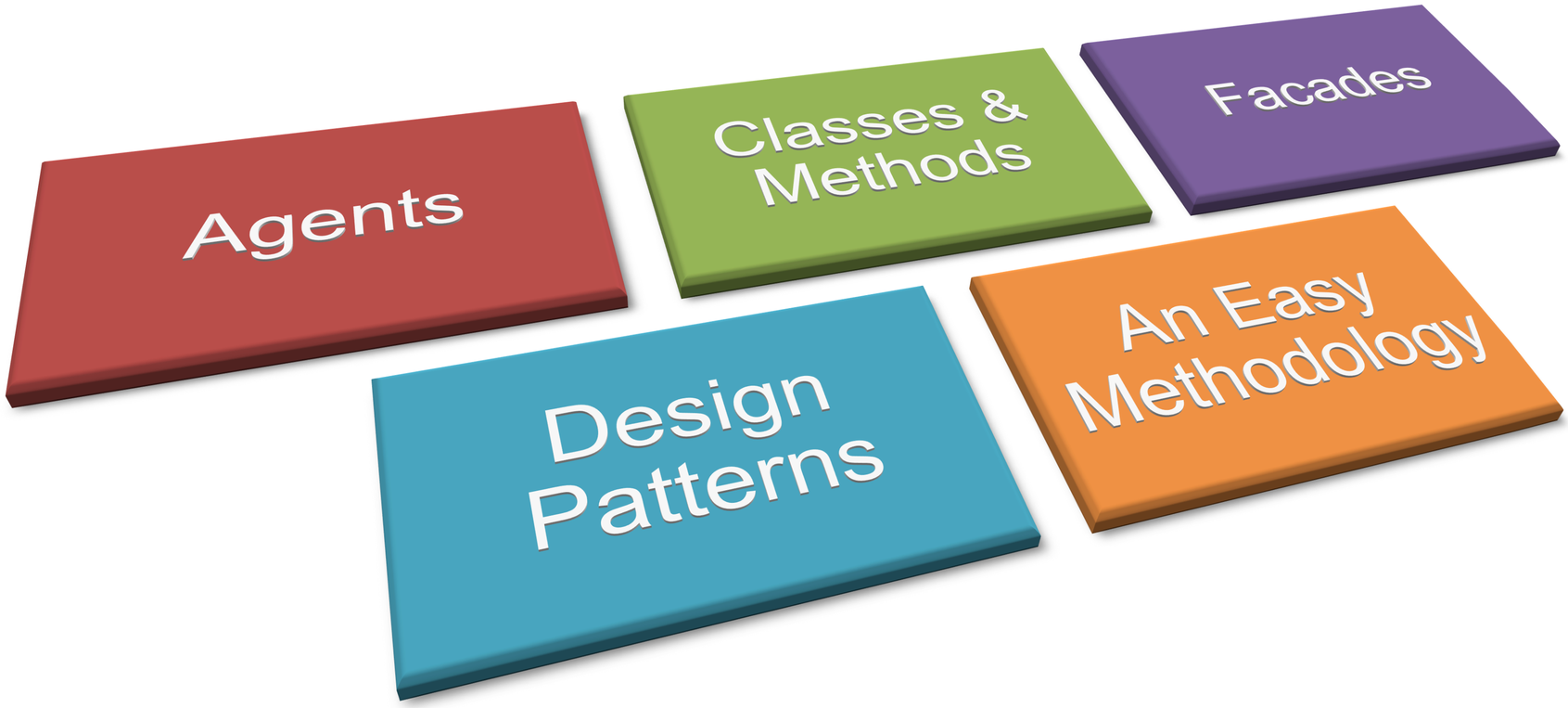

What Is Repeatable IP?
How Do I Write Repeatable IP?
How Do I Get Started?

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

AGENT

Medic

PROCESS

At his Office

On Home Visit

In Hospital

ACTION

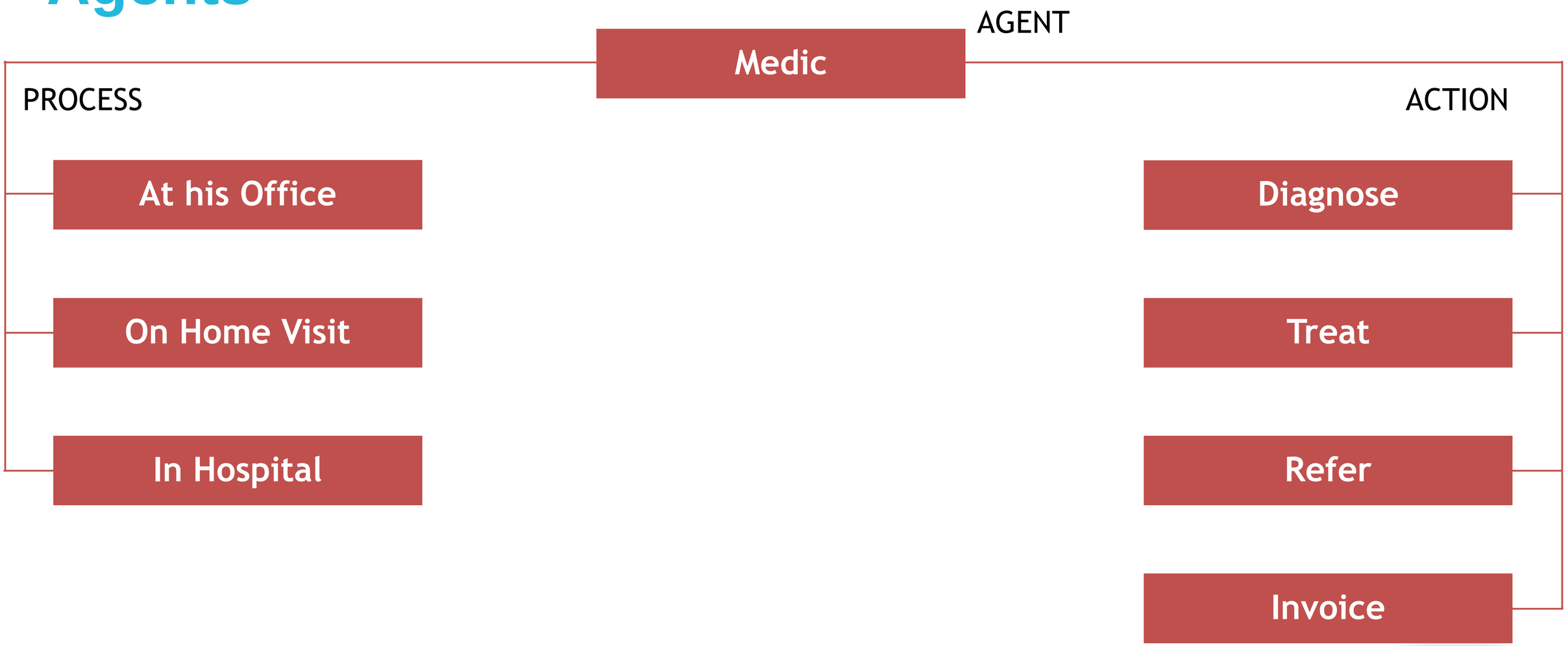
Diagnose

Treat

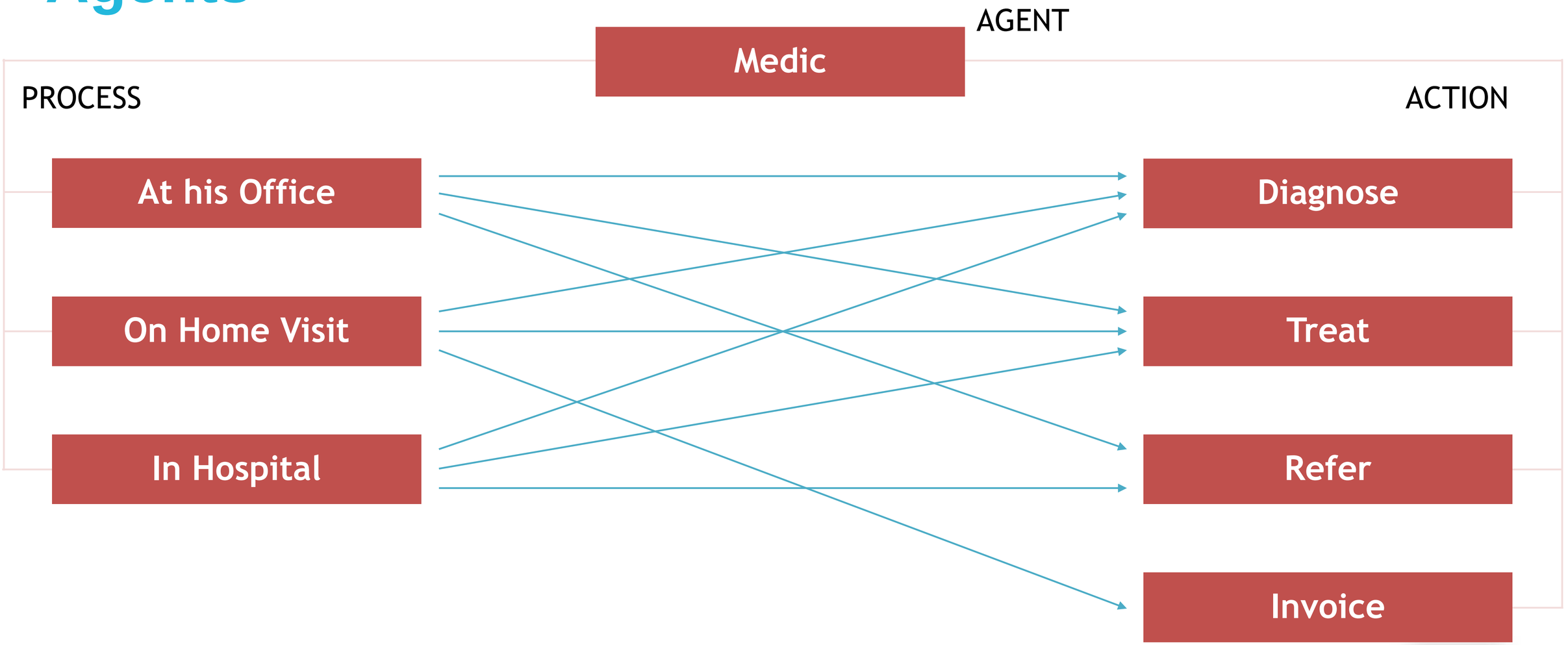
Refer

Invoice

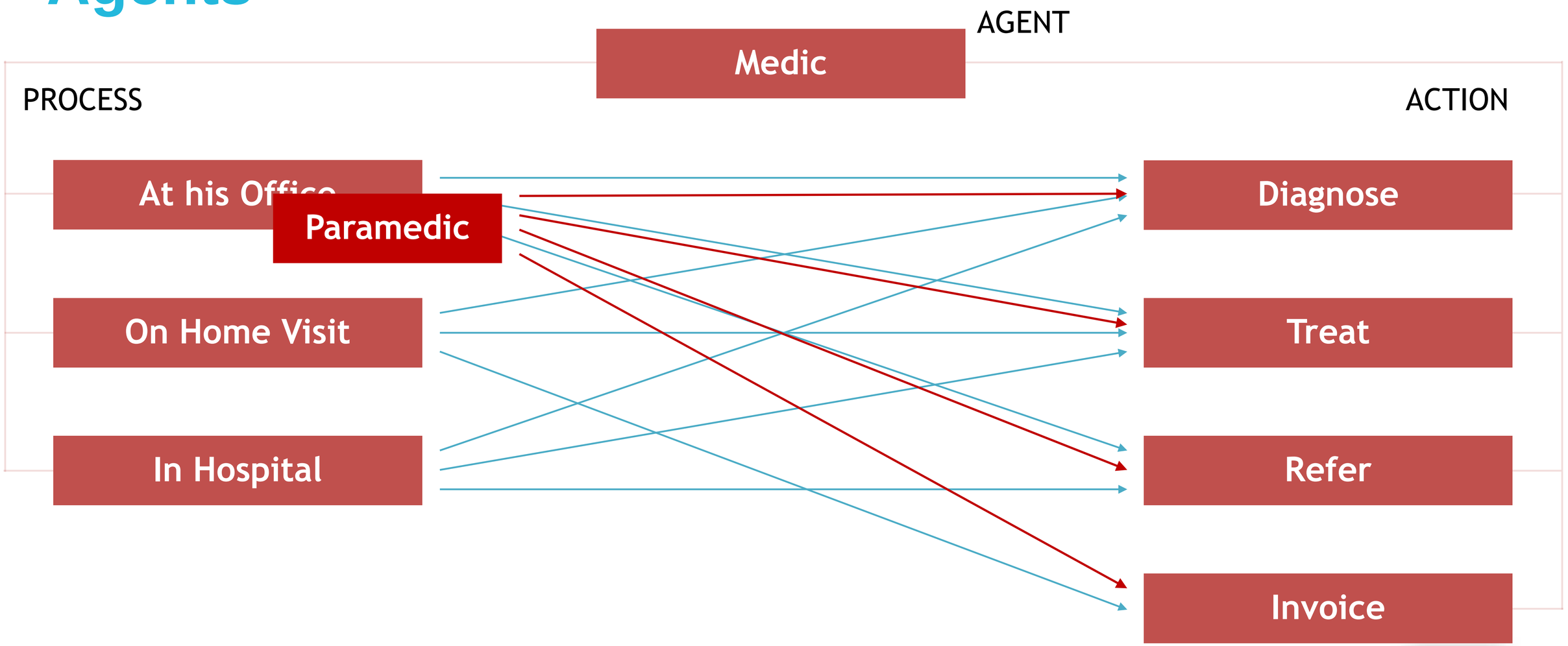
Agents


AGENT
Medic
PROCESS
ACTION
At his Office
On Home Visit
In Hospital
Diagnose
Treat
Refer
Invoice

Agents


AGENT
Medic
PROCESS
ACTION
At his Office
On Home Visit
In Hospital
Diagnose
Treat
Refer
Invoice

Agents


AGENT
Medic
PROCESS
ACTION
At his Office
Paramedic
On Home Visit
In Hospital
Diagnose
Treat
Refer
Invoice

Agents

Page 50 Purchase Order - C/AL Editor

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

Agents

Page 50 Purchase Order - C/AL Editor

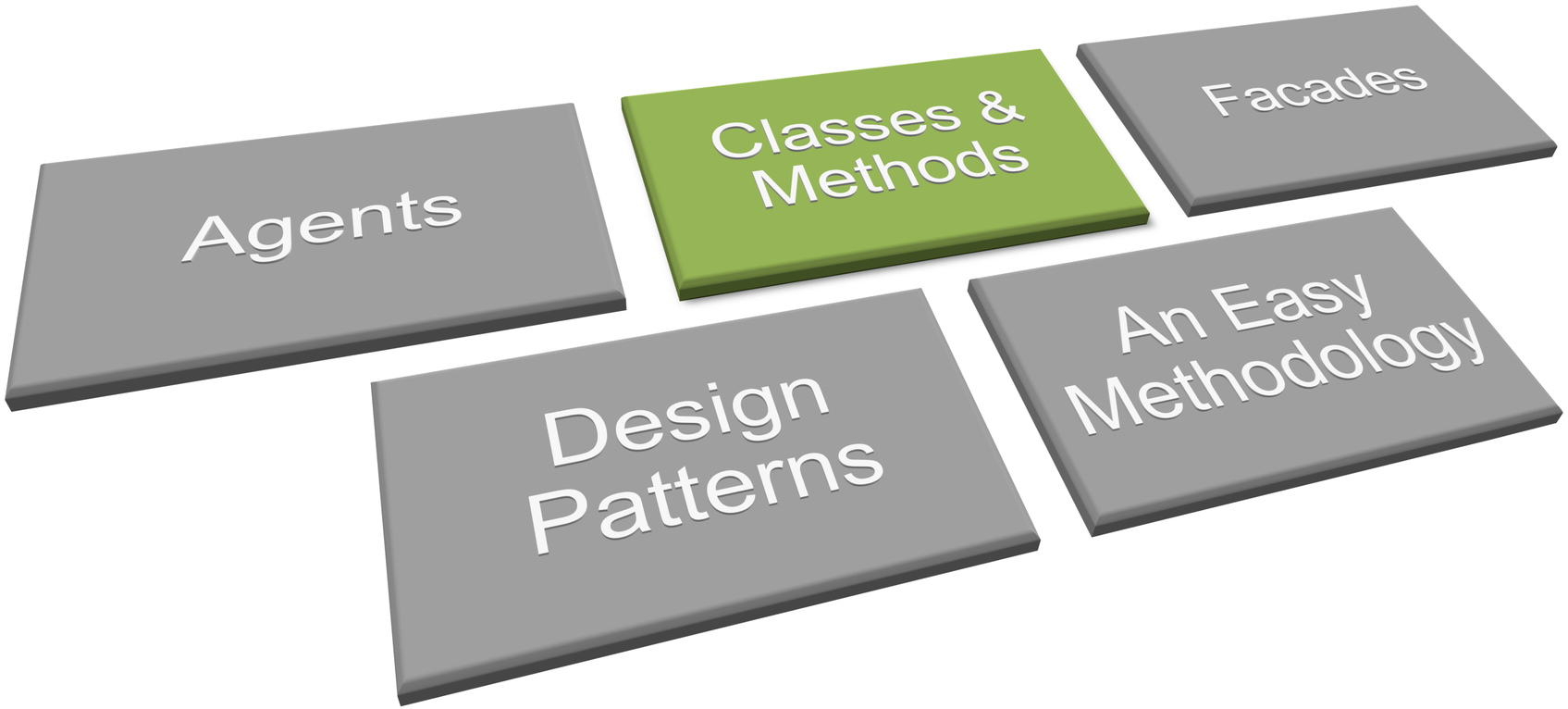
```
Diagnose - OnAction()
MedicManagement.ExecuteDiagnosis("Buy-from Vendor No.");
```

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)


Agents
Classes & Methods
Facades
Design Patterns
An Easy Methodology

Classes & Methods

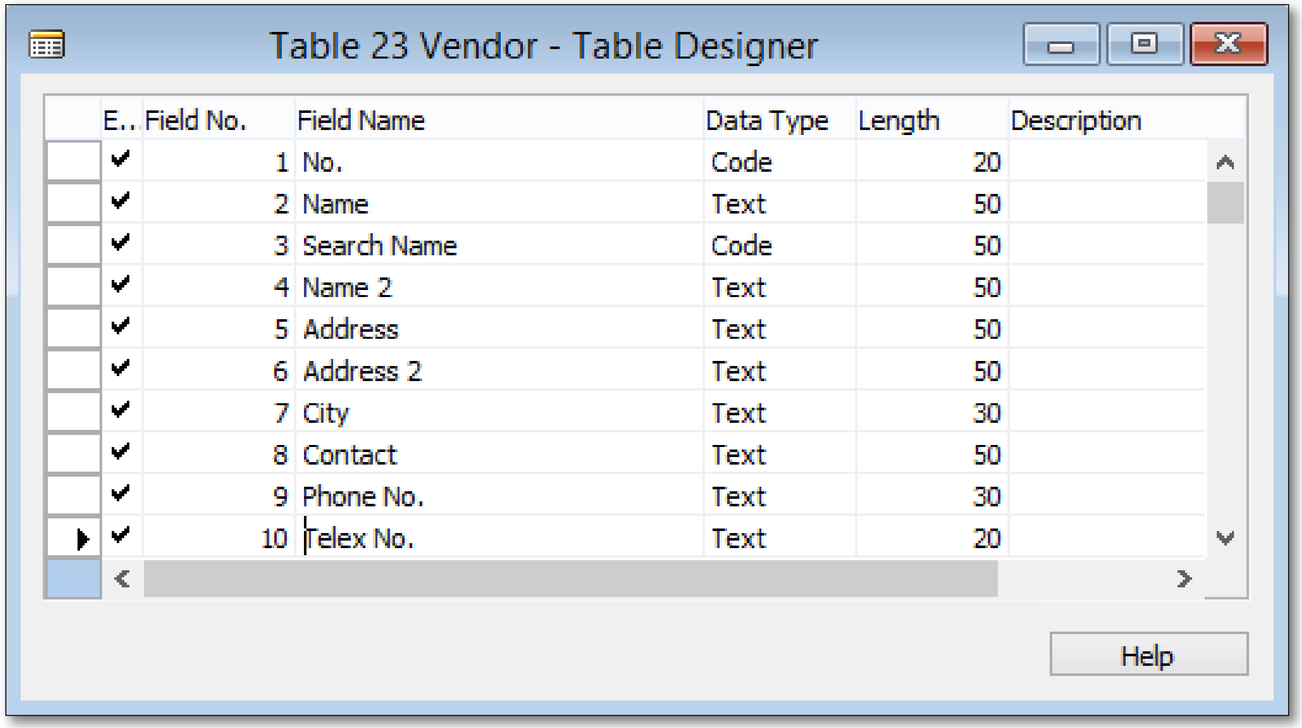
- Classes are tables
- They carry all attributes + methods
- Attributes are fields

Methods are functions

All methods are always declared on the class

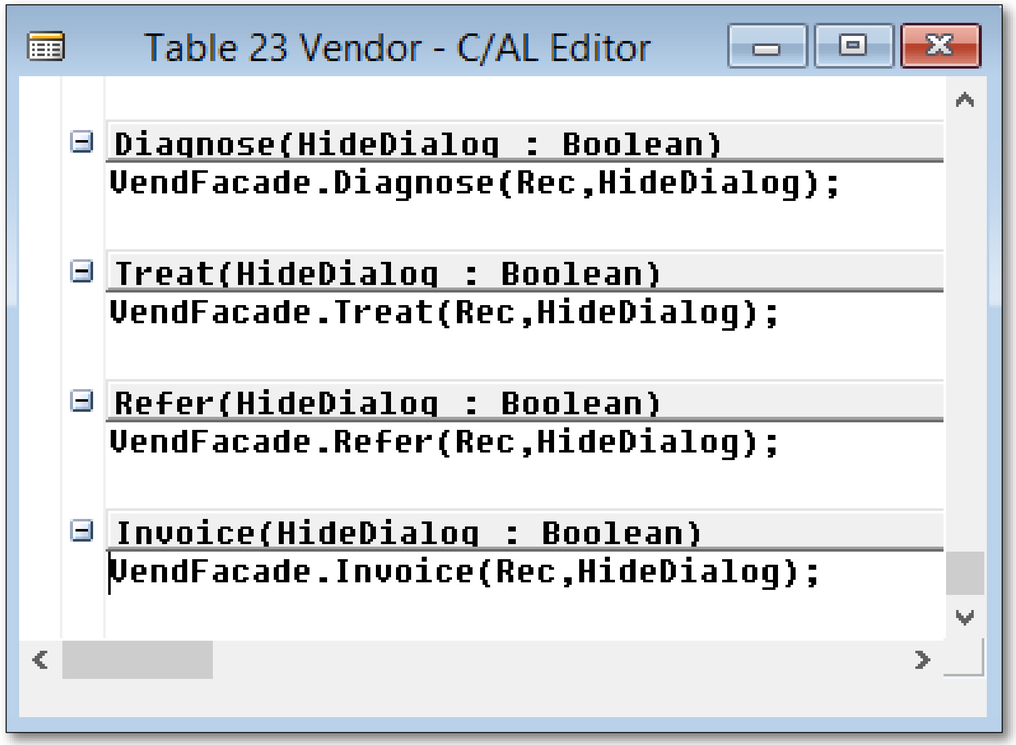
Classes & Methods

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



Classes & Methods

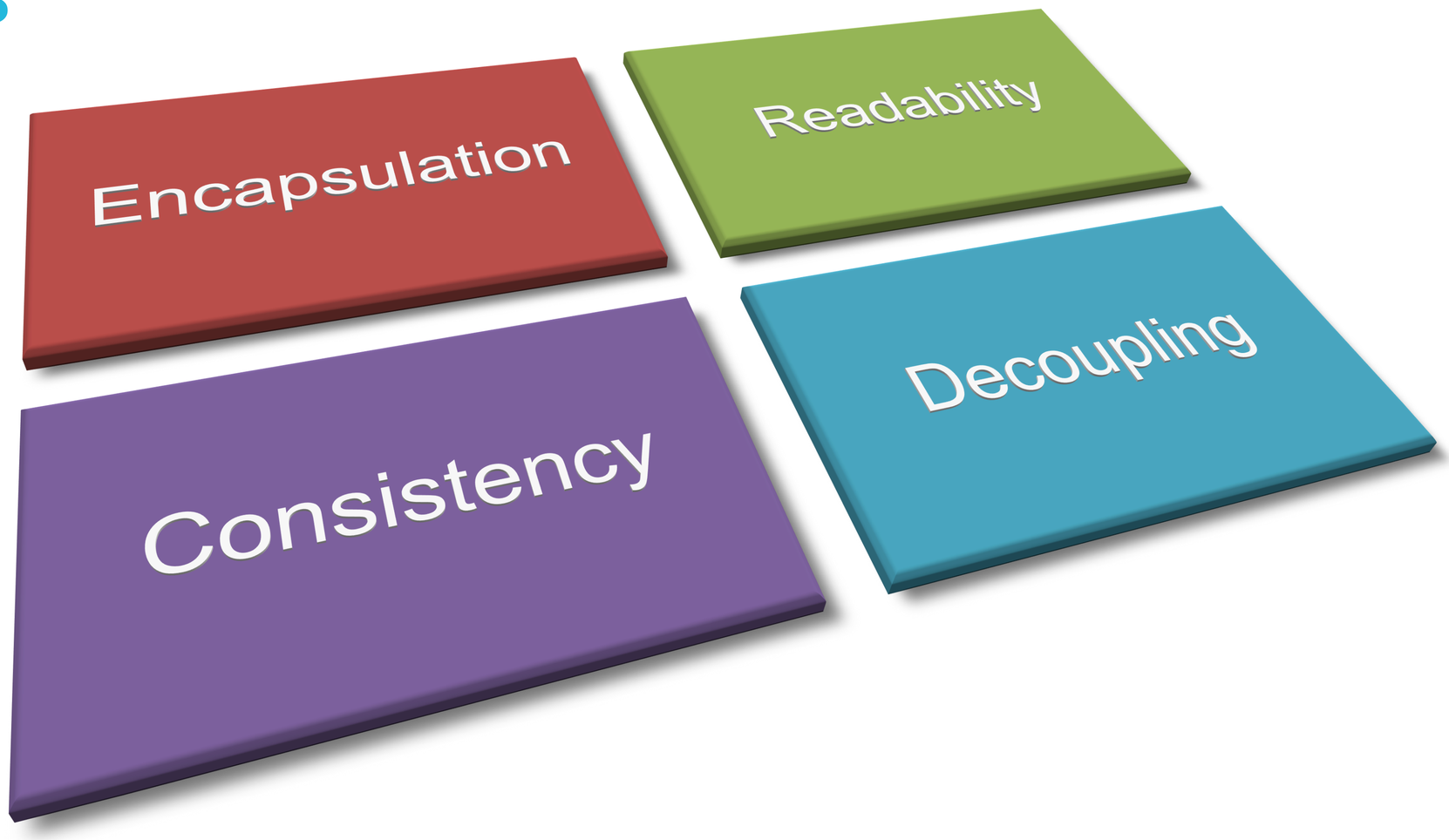
- All methods (functions) are declared on the class (table)



Method Declaration: Advantages

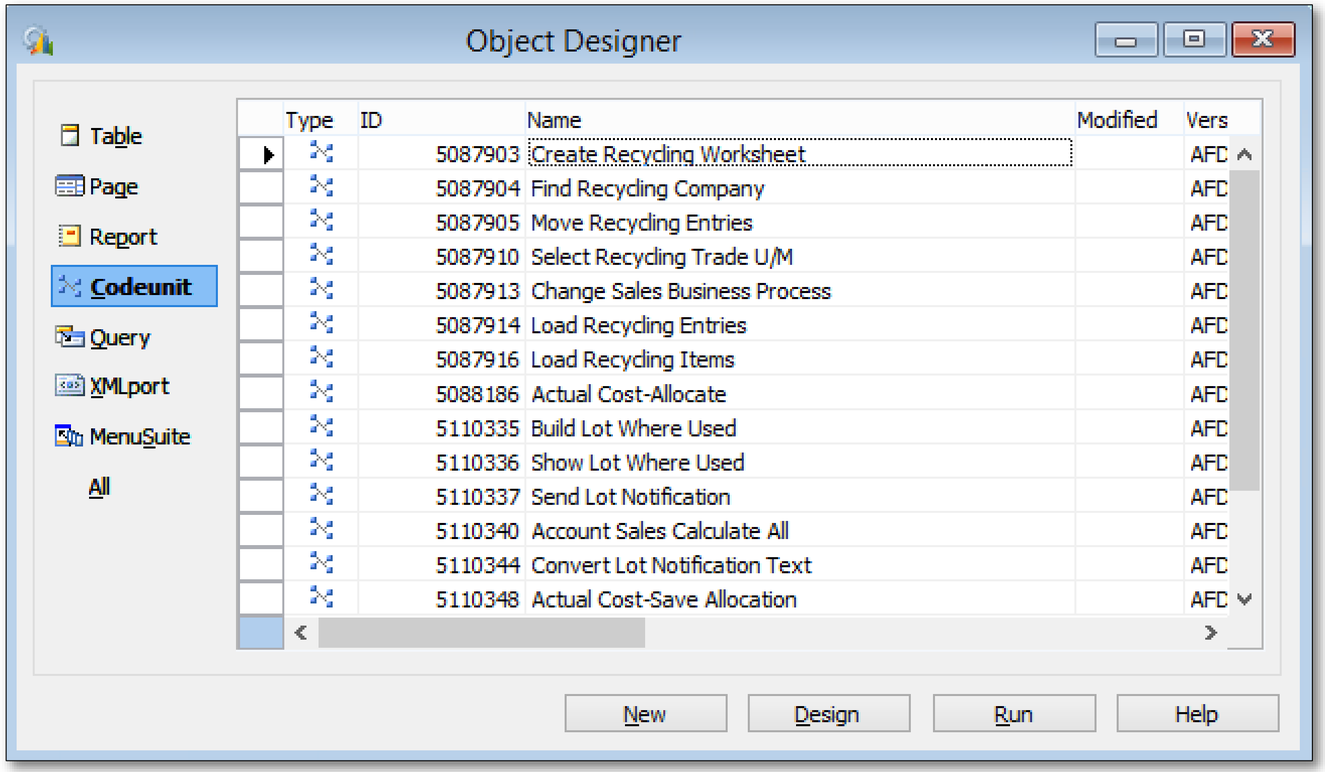
- You see explicitely …
  - which functions exist for each table
  - which methods exist for each class
  - which actions are defined for each agent
- IntelliSense (F5)
- But …
  - methods should not be coded on the class

Methods



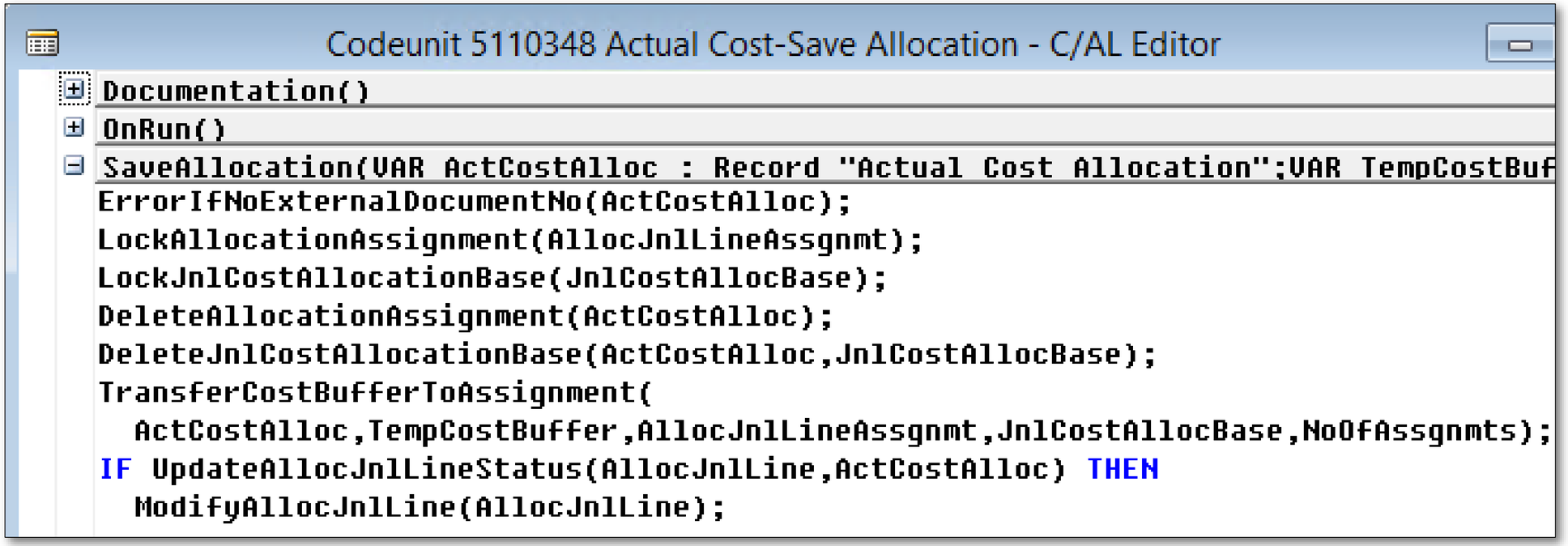
Encapsulation

- Each method is a codeunit of its own



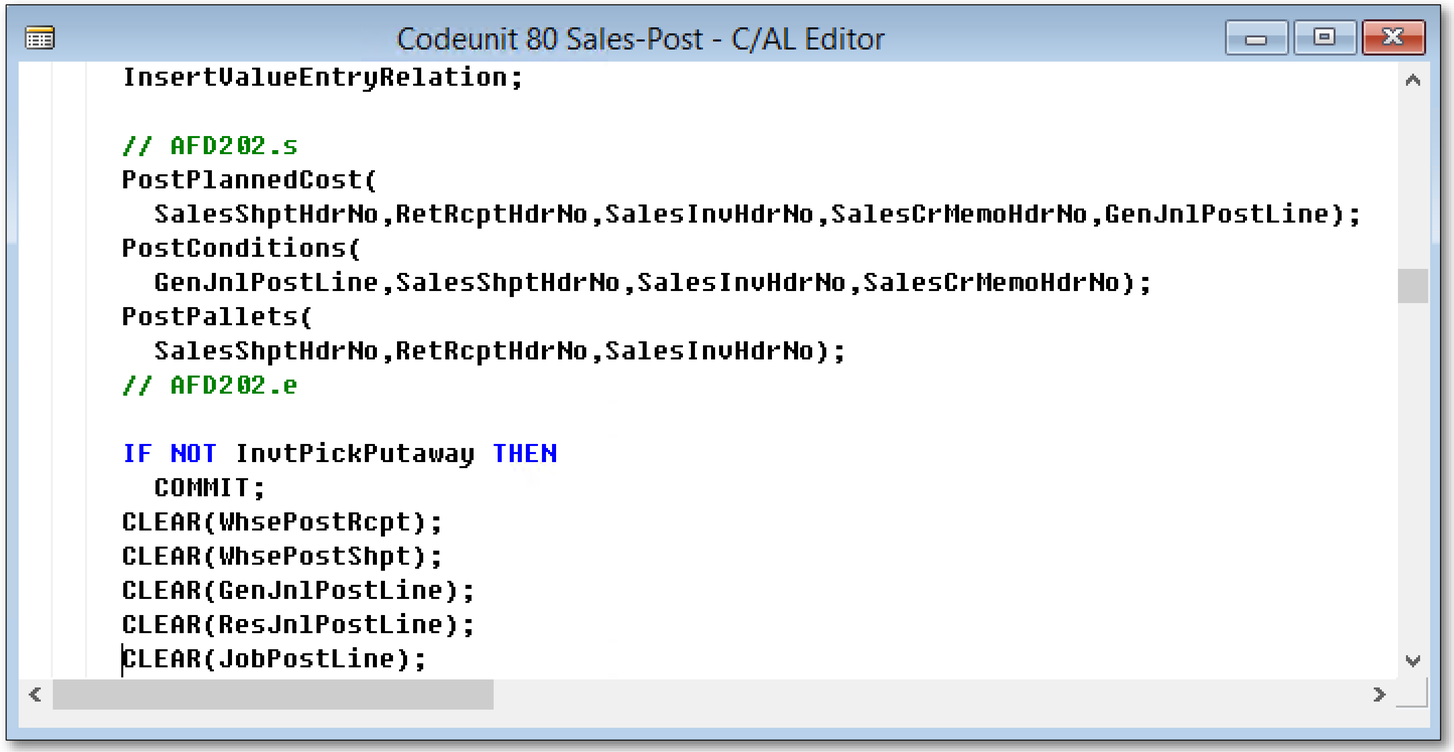
Encapsulation

- Each method only has one global function



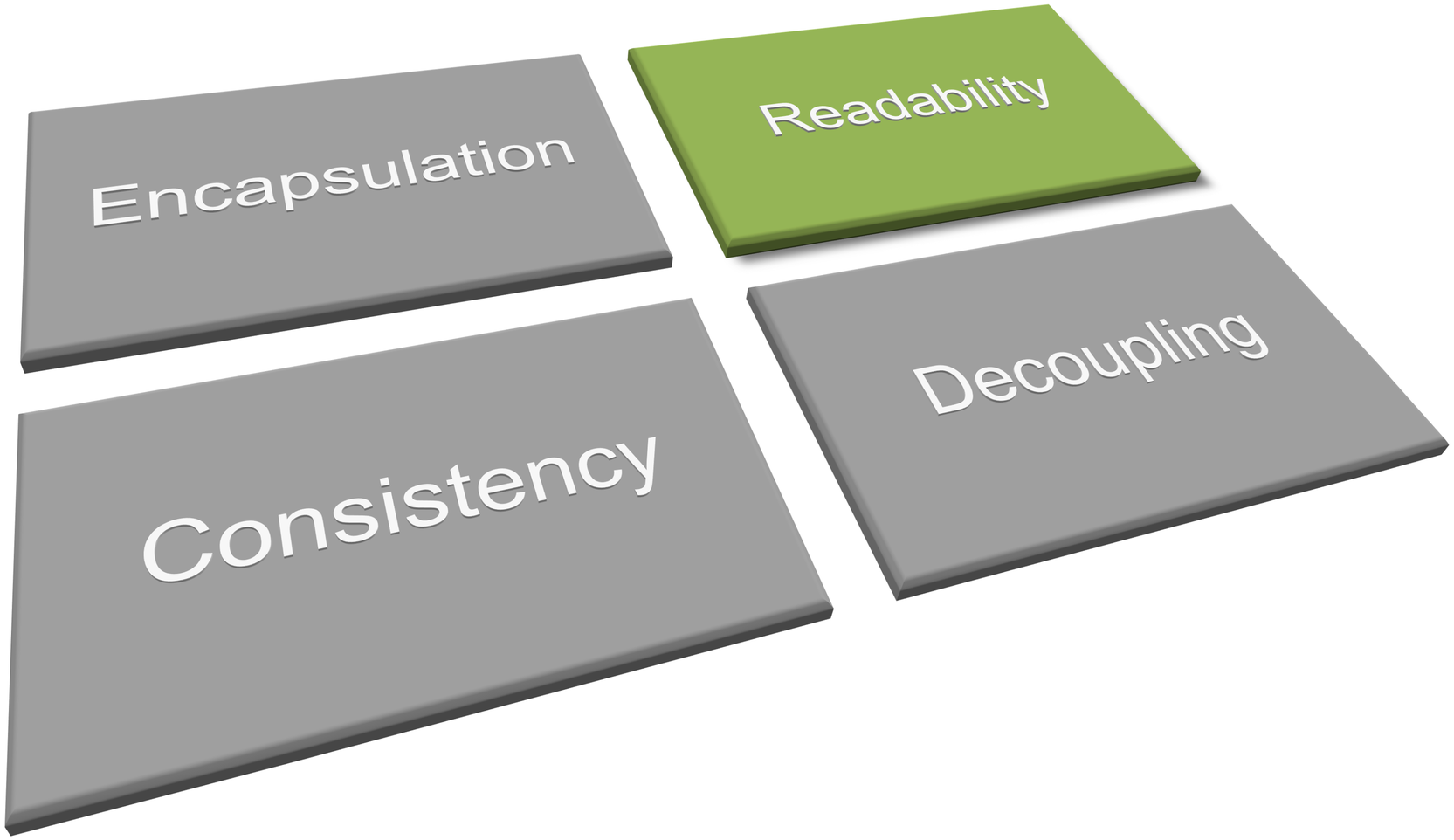
Encapsulation

- A method is always instantiated through its class



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

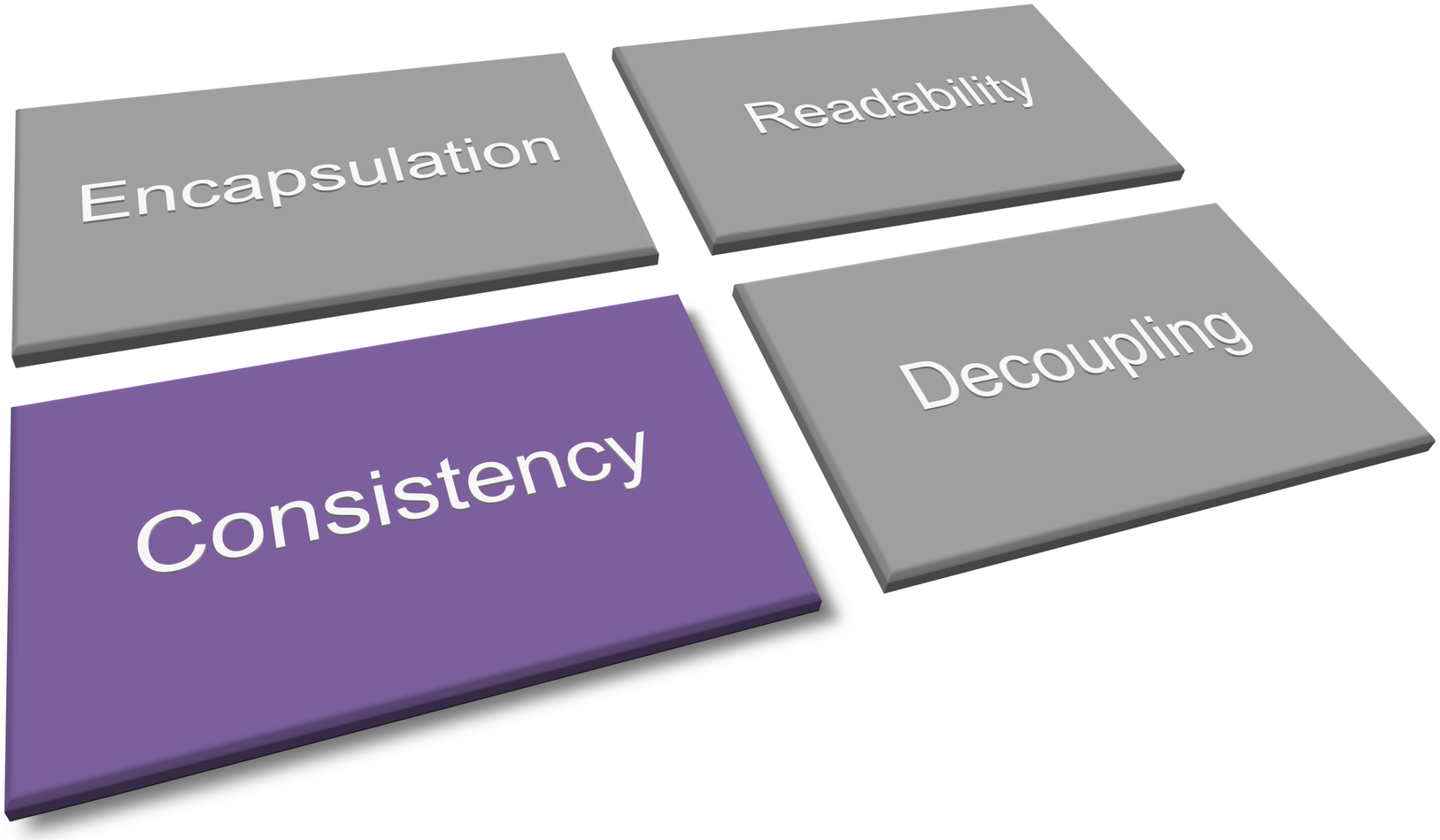
- The global function of each method must be a readable flow chart of what the method does
- It should include all relevant steps
- Natural Language Code: It should resemble understandable plain text
- It must not contain any “nerdy” stuff
  - Cust.GET is nerdy
  - GetCustomer is not

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(
```

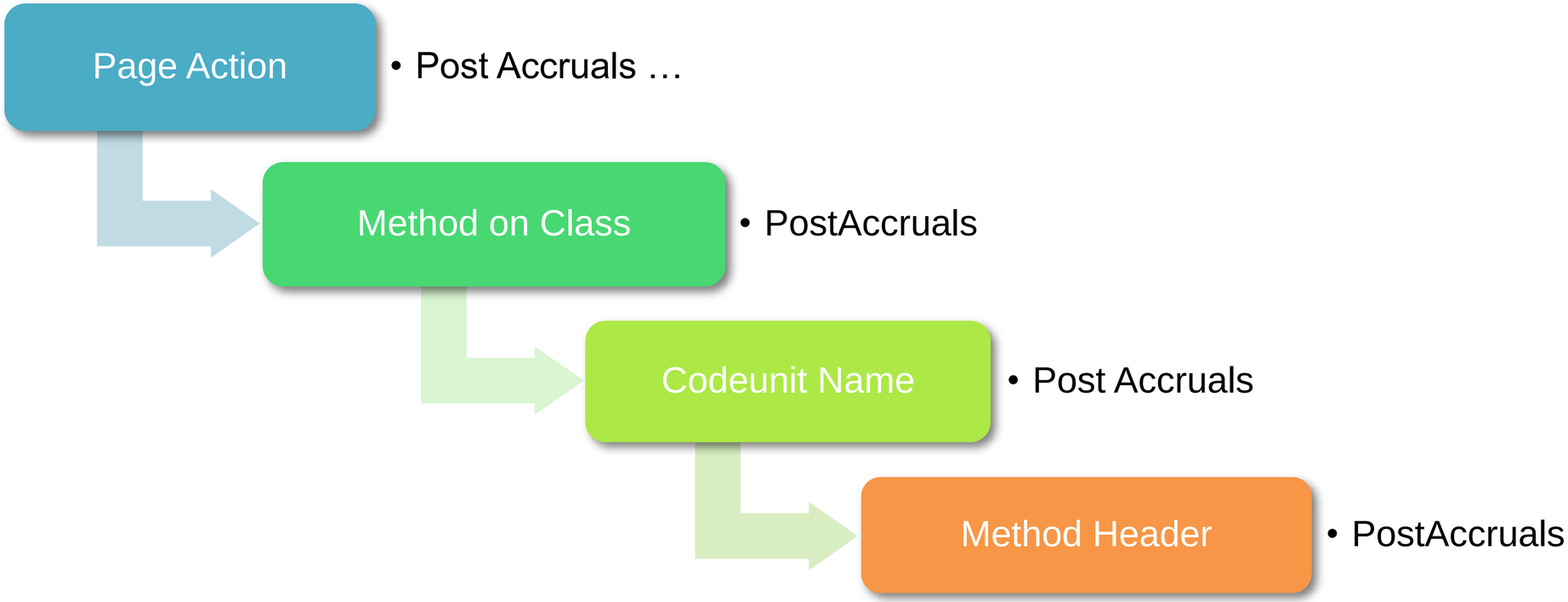
Process Components

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


Encapsulation
Readability
Consistency
Decoupling

Consistency

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



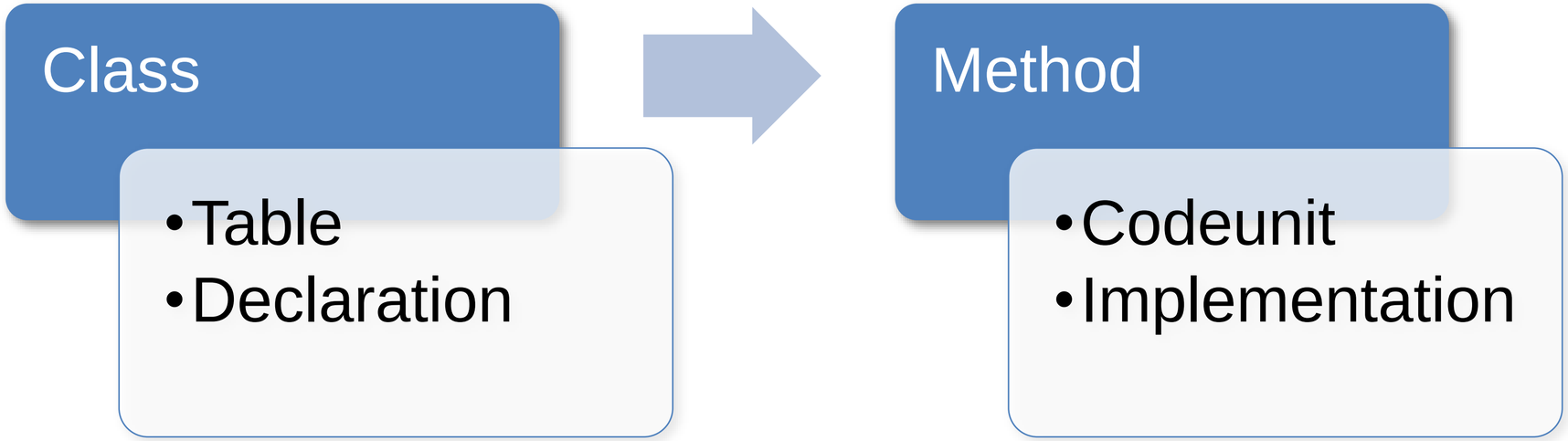
Extending Consistency

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

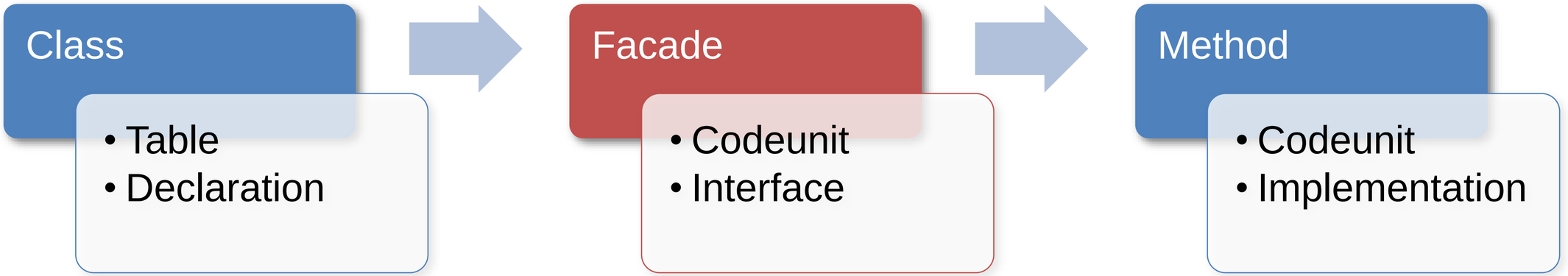

Encapsulation
Readability
Consistency
Decoupling

Facade

Decoupling: Facade



Decoupling: Facade

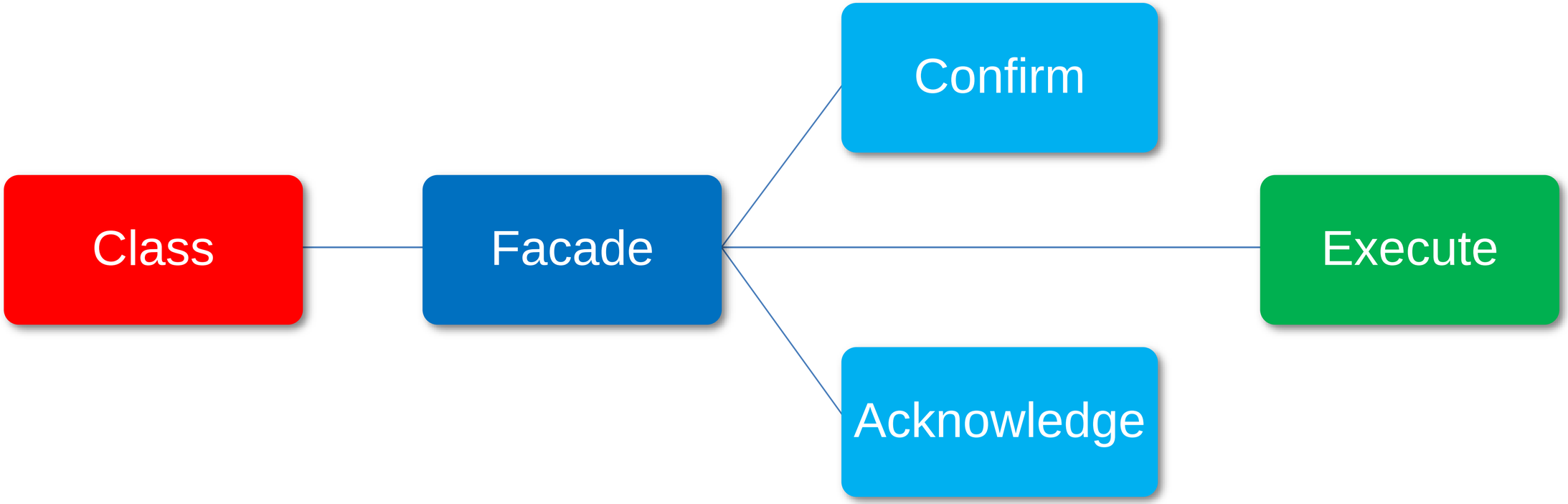


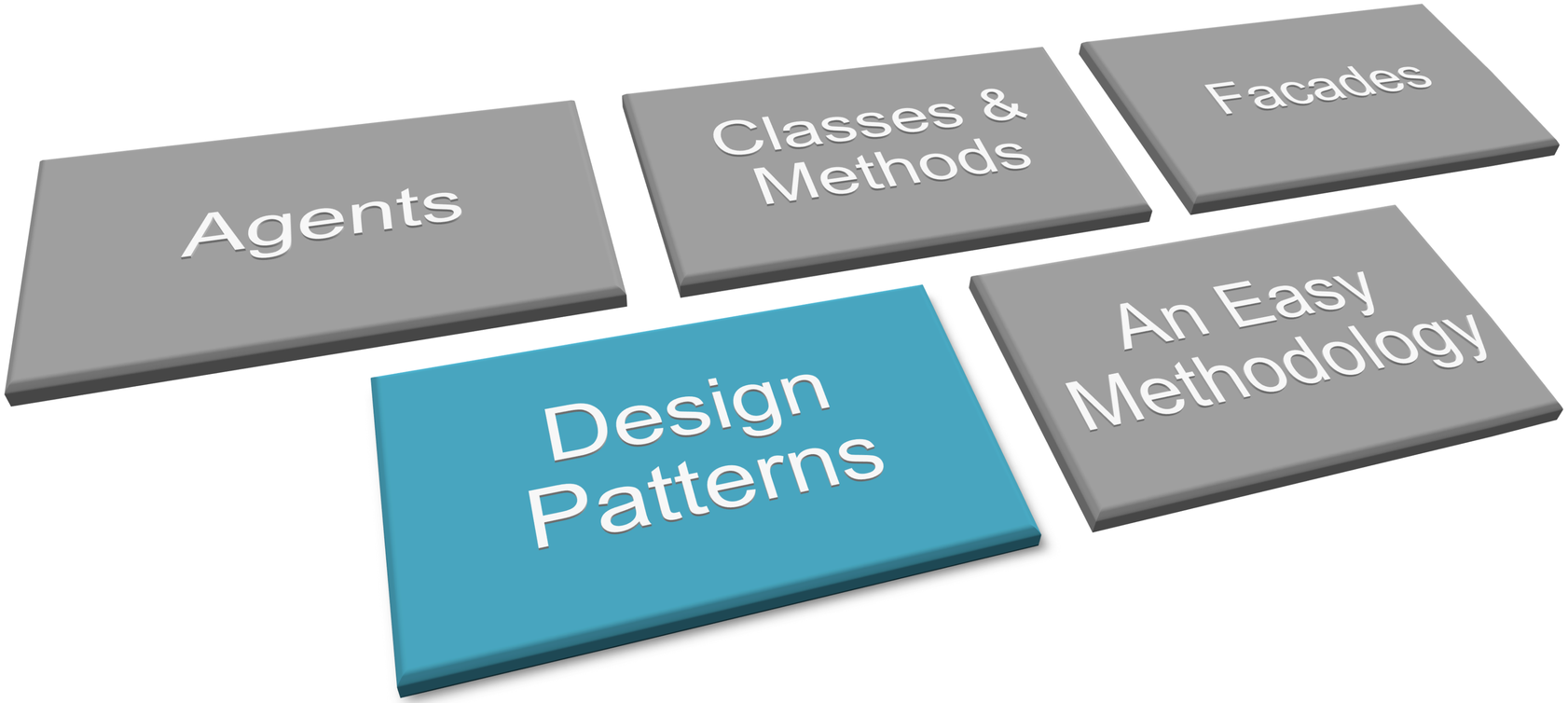
Facade

Codeunit 5088183 Actual Cost Allocation Facade - C/AL Editor

```
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




Agents
Classes & Methods
Facades
Design Patterns
An Easy Methodology

Design Patterns

- A facade is a Design Pattern

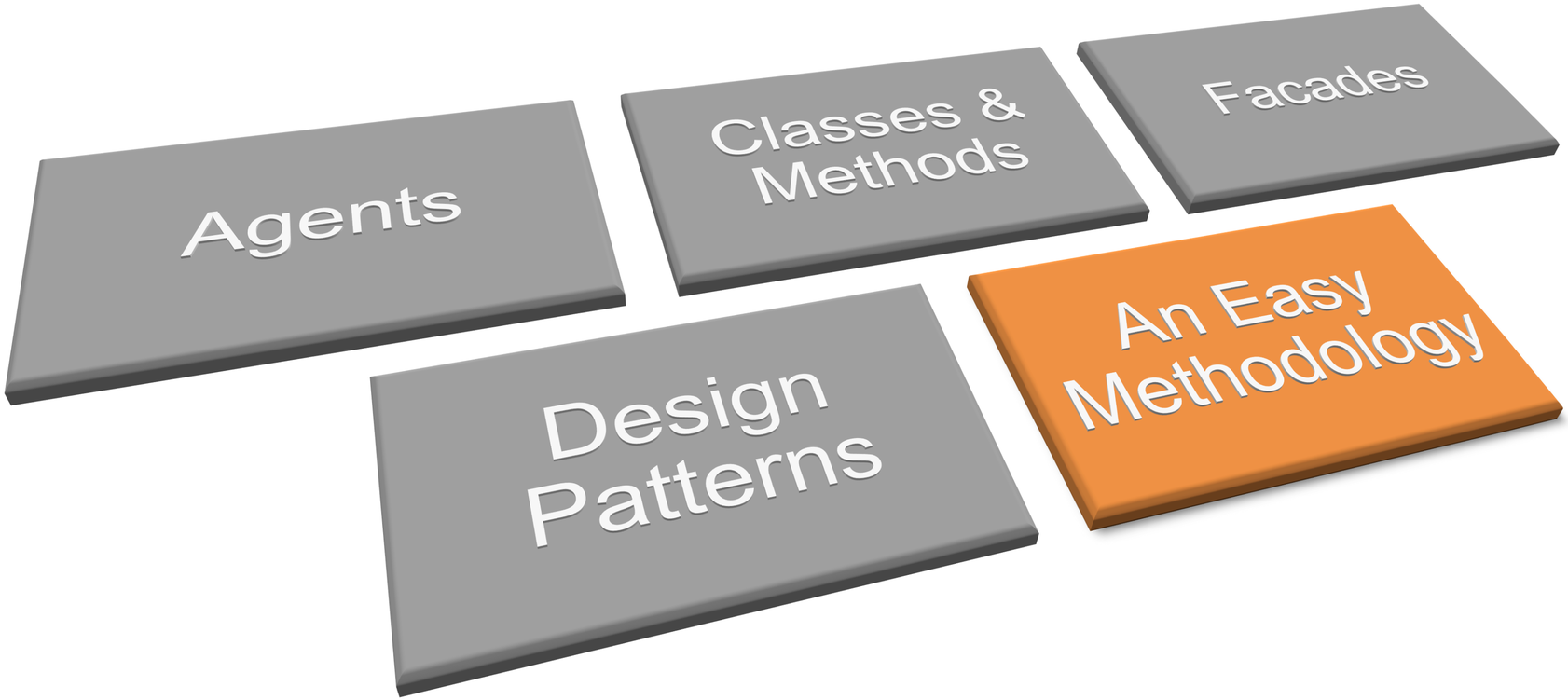
Design Patterns have always been the „secret source“ of NAV, enabling developers to learn new areas of the application and be productive without a huge ramp up. This makes design patterns key to understanding the architecture of NAV and writing repeatable code.

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


Agents
Classes & Methods
Facades
Design Patterns
An Easy Methodology

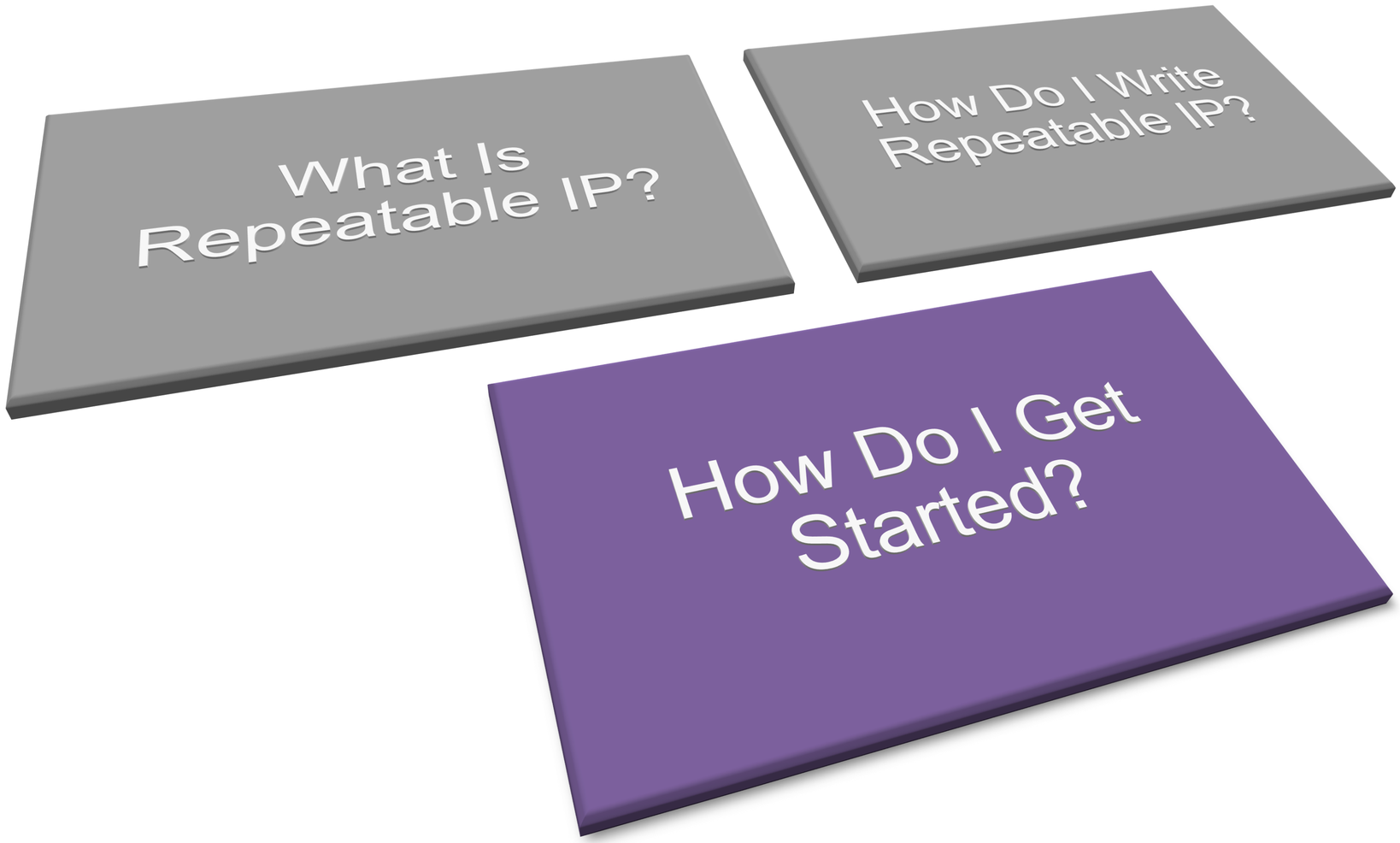
An Easy Methodology

- 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.

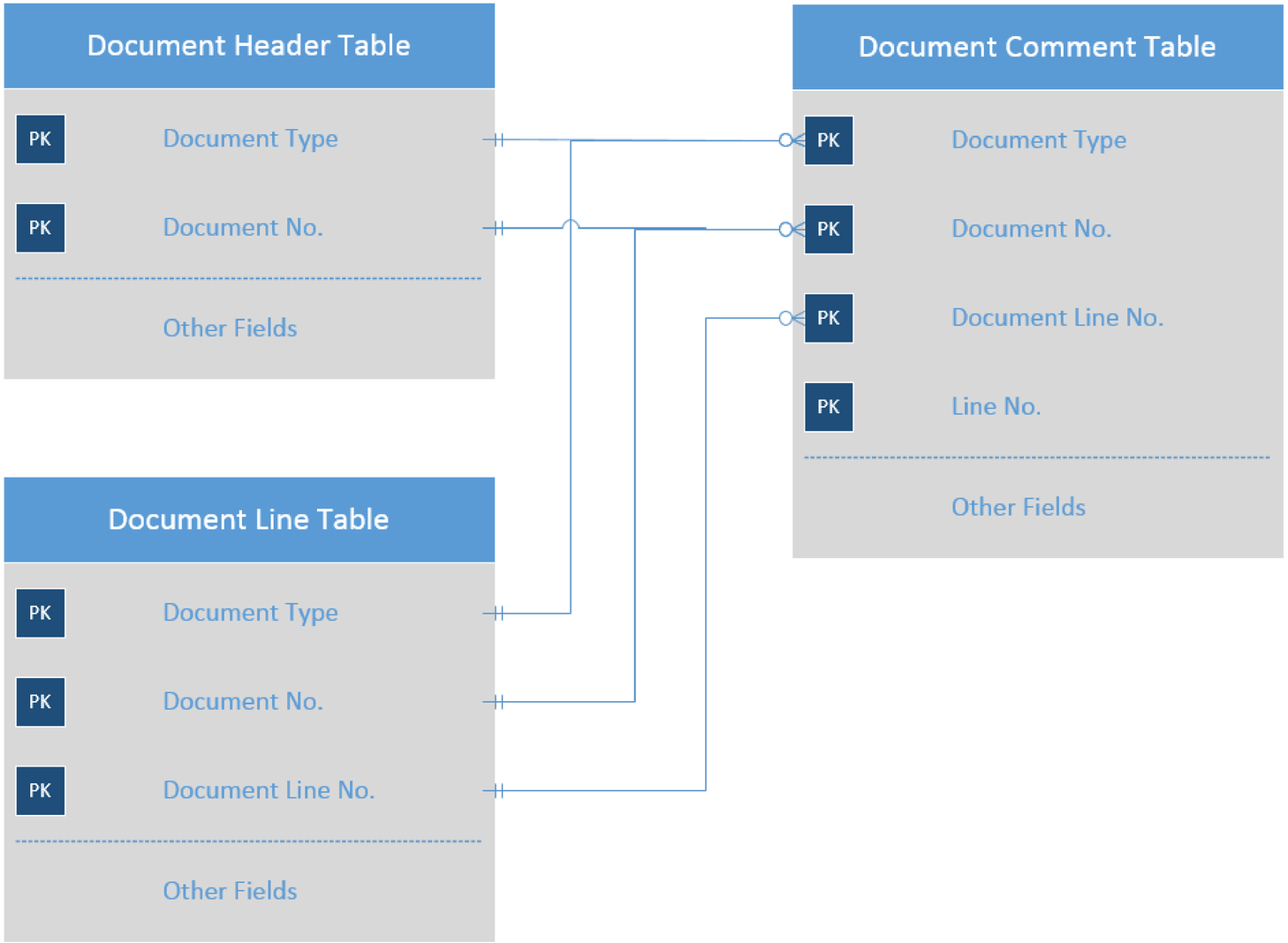
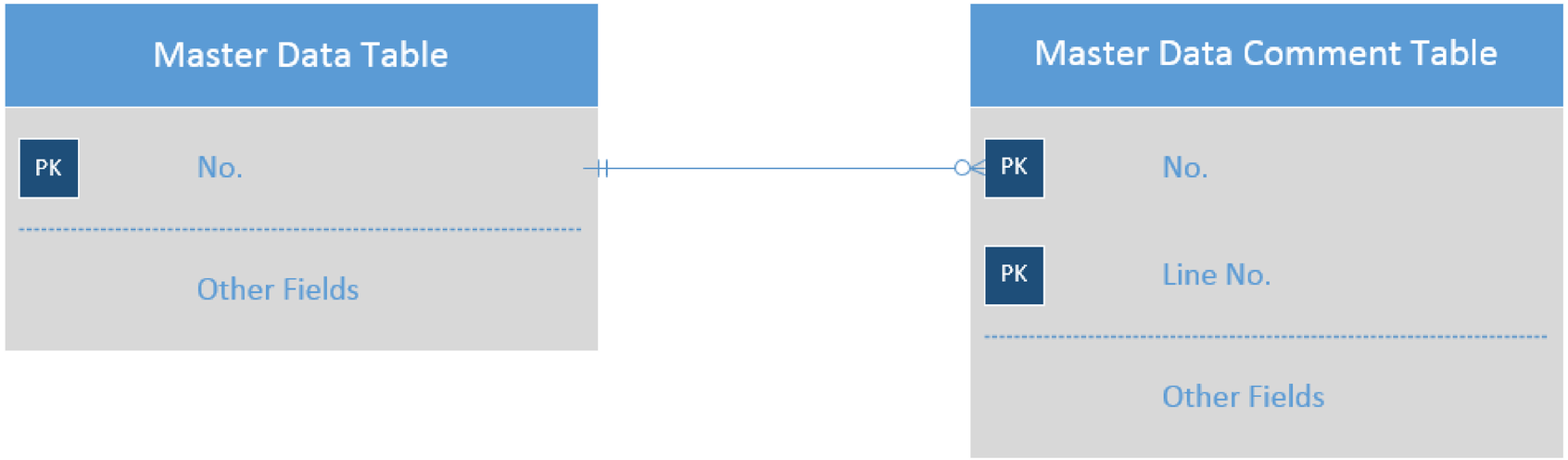
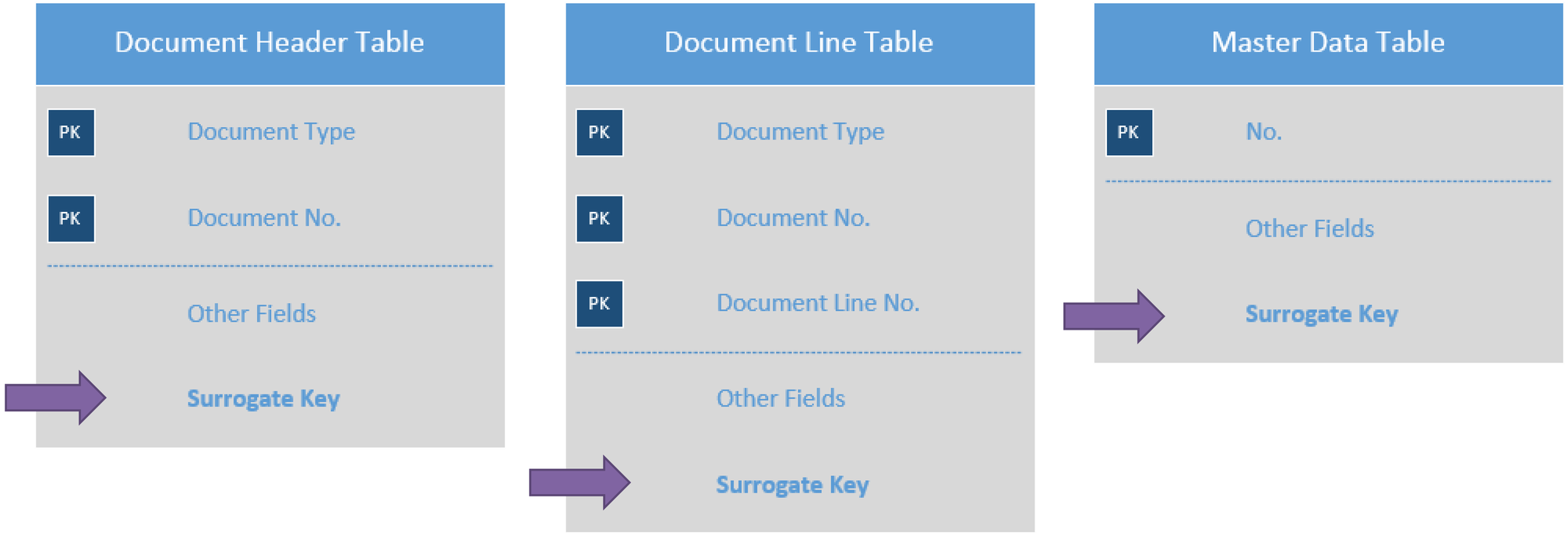


Table Structure for linking a Master Data Table with a Master Data Comment Table



How to do this with a Surrogate Key!

Step 1: Adding the Surrogate Key

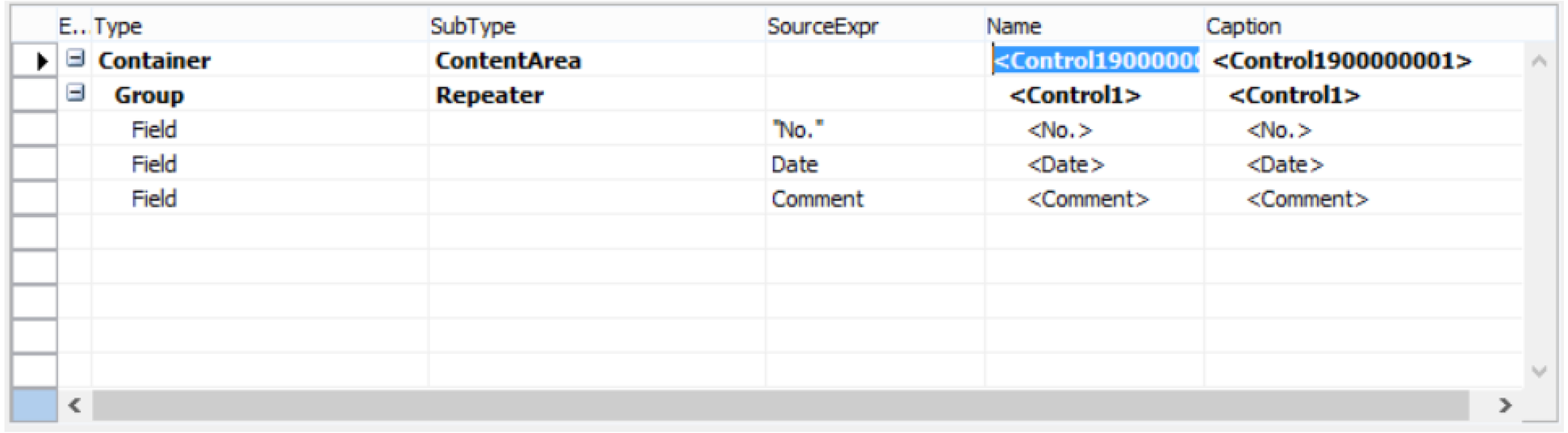


Step 2: Create a generic link-able sub table

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.

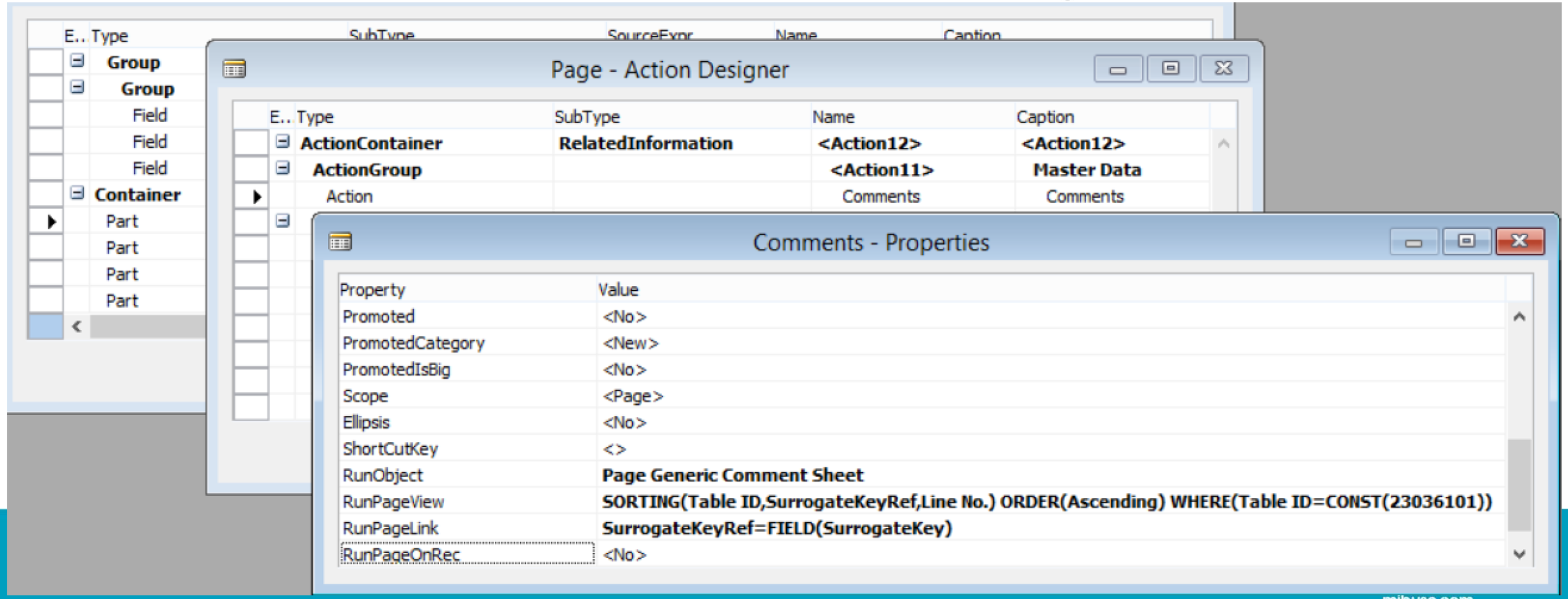


E..	Type	SubType	SourceExpr	Name	Caption
▶ ⊟	**Container**	**ContentArea**		**<Control1900000(**	**<Control1900000001>**
⊟	**Group**	**Repeater**		**<Control1>**	**<Control1>**
	Field		"No."	<No.>	<No.>
	Field		Date	<Date>	<Date>
	Field		Comment	<Comment>	<Comment>

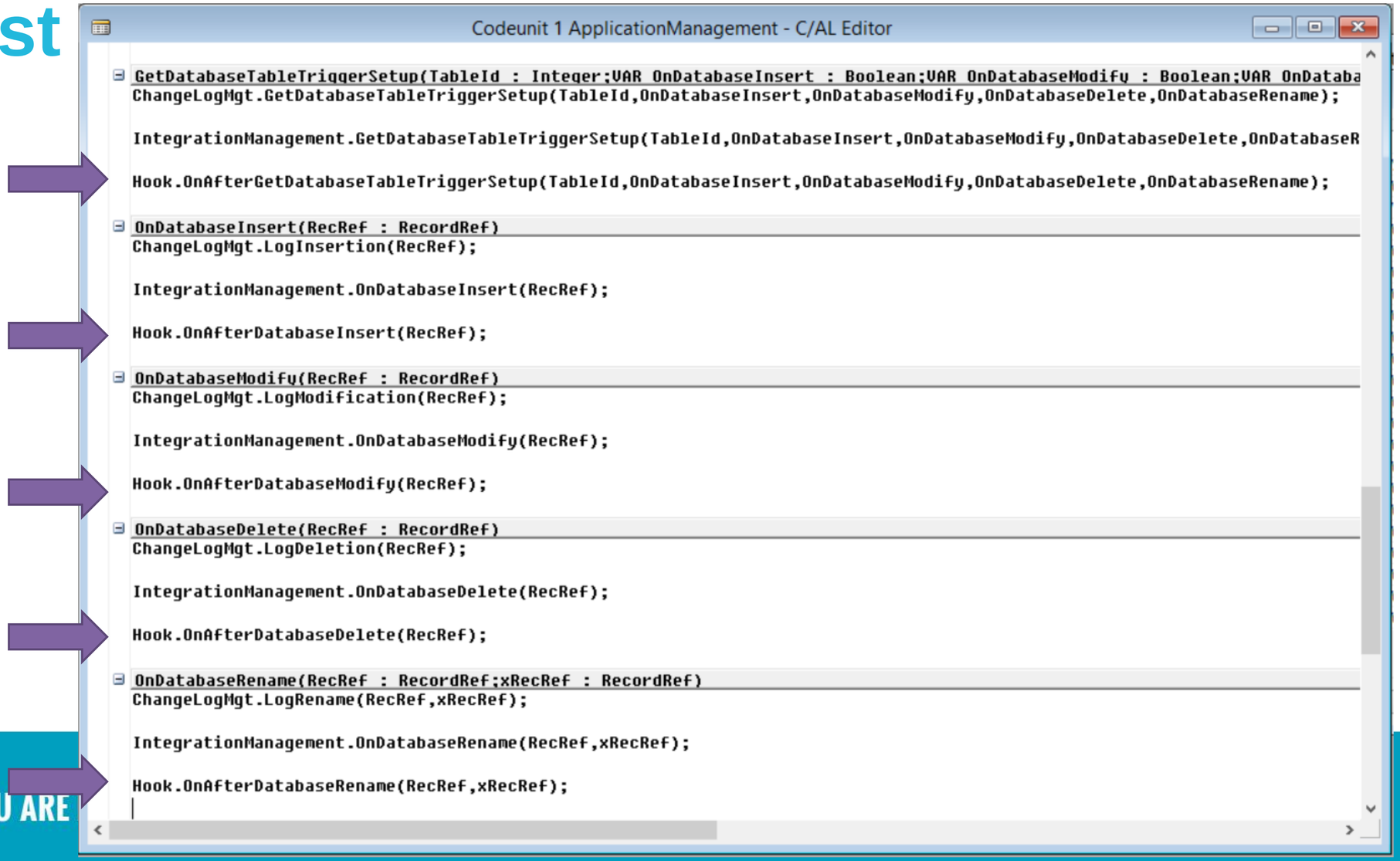
This will look exactly like a standard Comment List and Sheet today!

Step 4: Add an Action

This could also be an Factbox or SubPage.



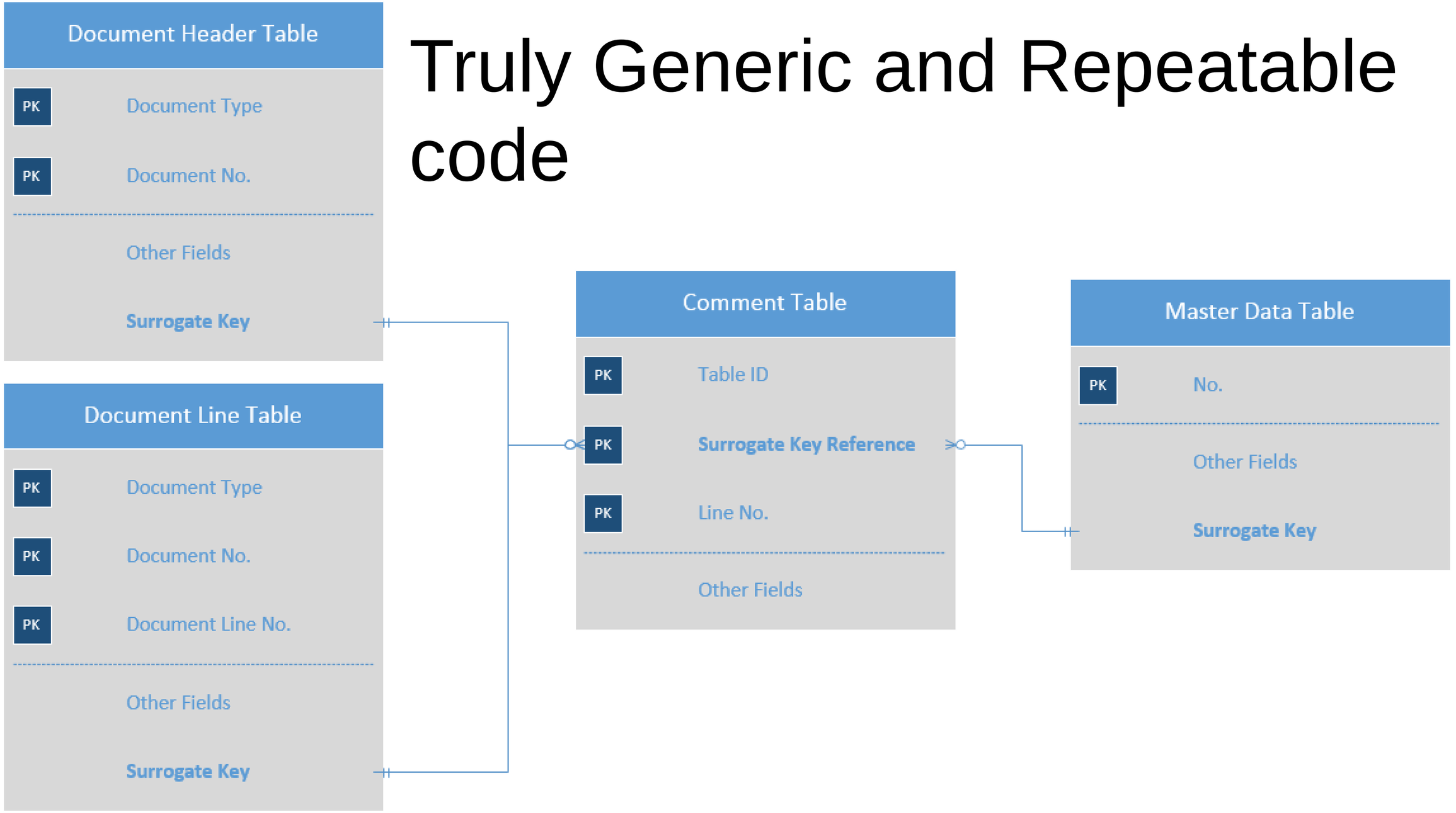
Step 5: Create a Hook for Functions in Codeunit 1 ApplicationManagement if one doesn't already exist



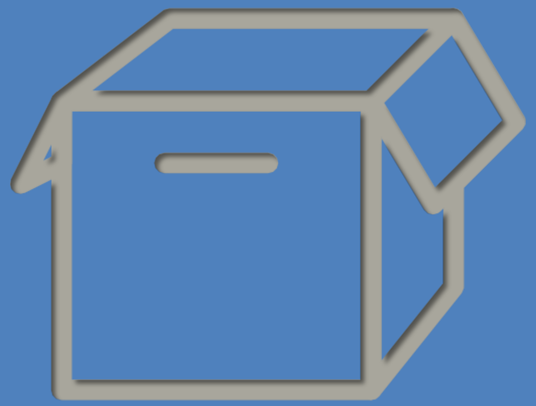
Step 6: Create a code to delete records in the Sub table if a main table record is deleted if this is required.

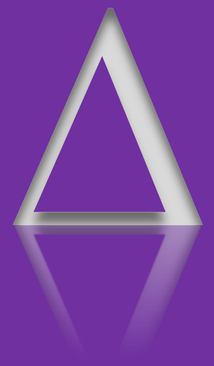
This function needs to be called from the Hook created in Step 5.

Truly Generic and Repeatable code



Out of the Box

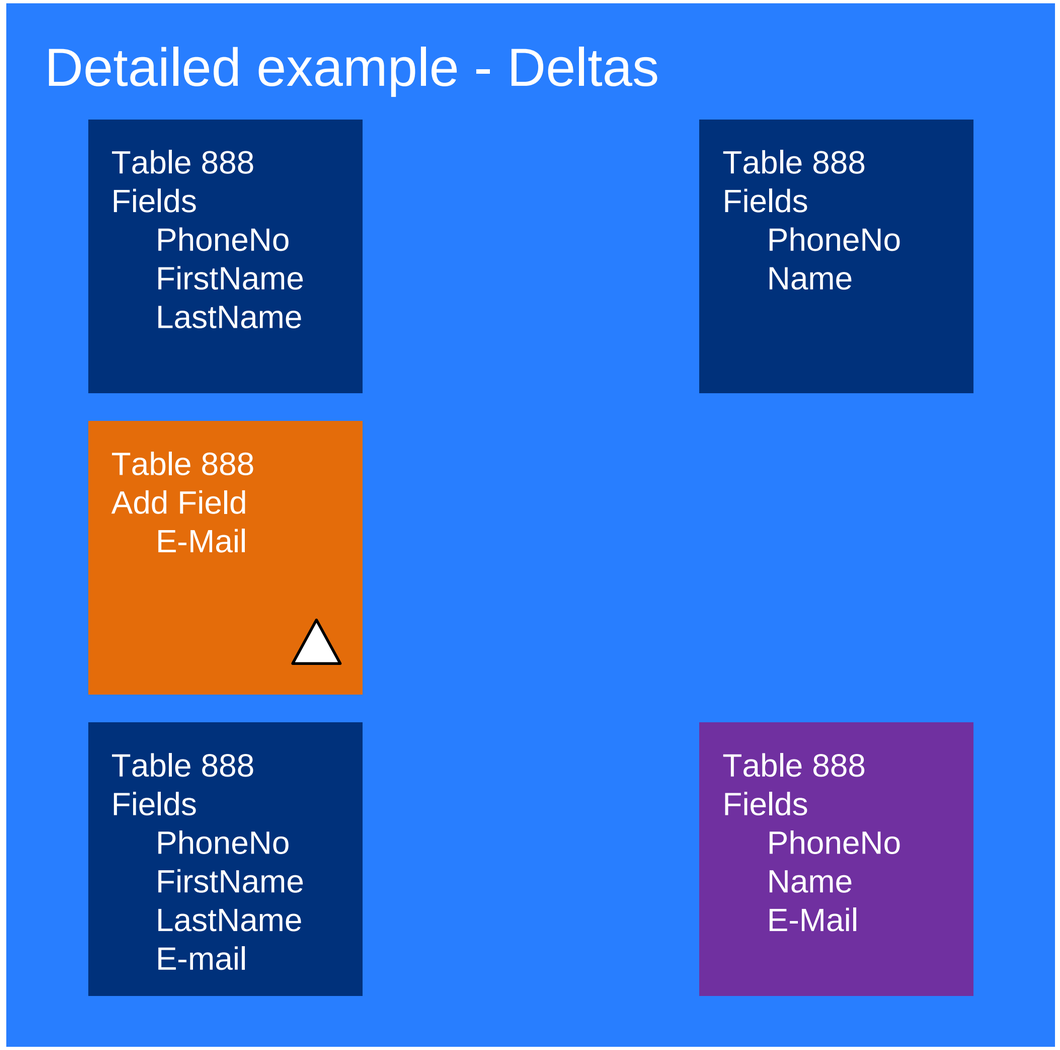




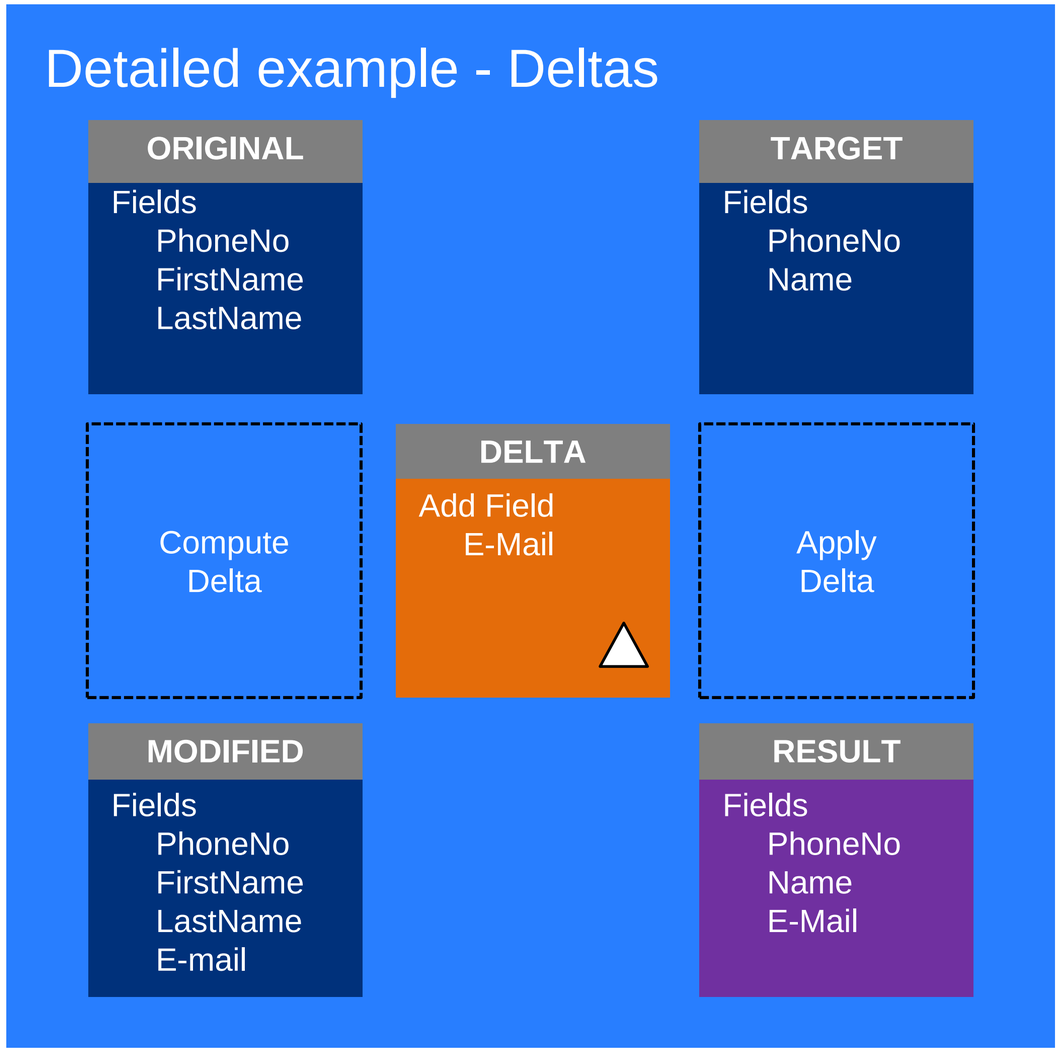
Shipping Delta’s

Power Shell | Merge

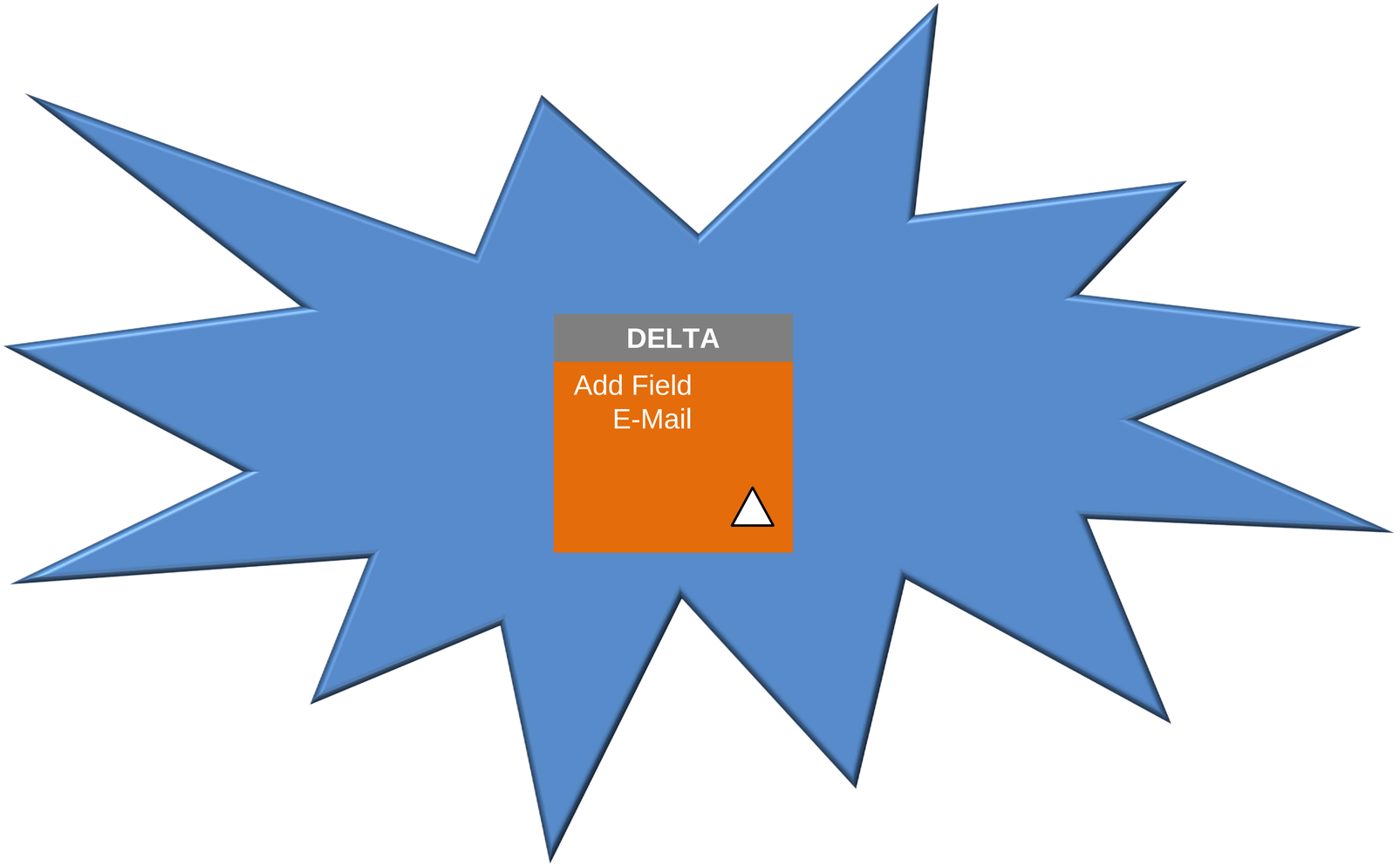
Detailed example - Deltas



Power Shell | Shipping Software



Power Shell | A New Star



SHIPPING SOFTWARE VIA DELTA

GOOD	BAD	UGLY
➢ Save Time ➢ Applies to all Databases	➢ Version List ➢ Conflicts are hard to handle	➢ Tables and Fields require Partner License

MORE TOMORROW…

Eric Wauters

BELGIUM

iFacto Business Solutions

09:00 - 10:30

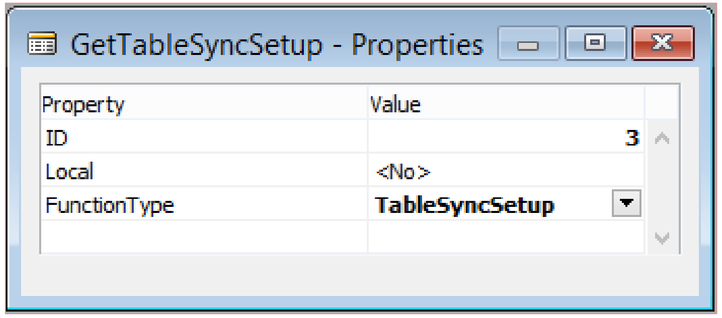
Looking at NAV the Powershell way …

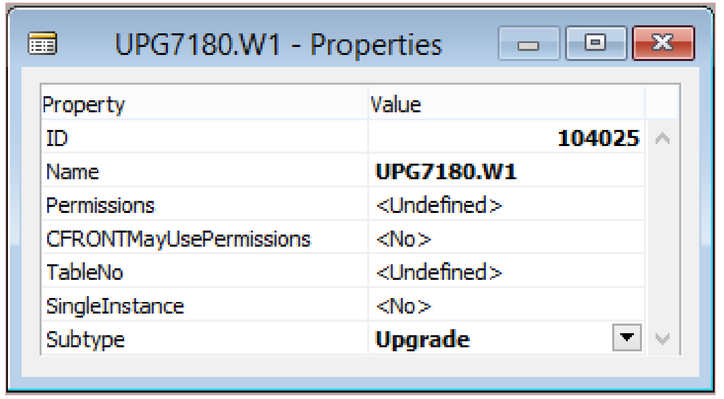
LEVEL: 200 - INTERMEDIATE | LEVEL: 300 - ADVANCED

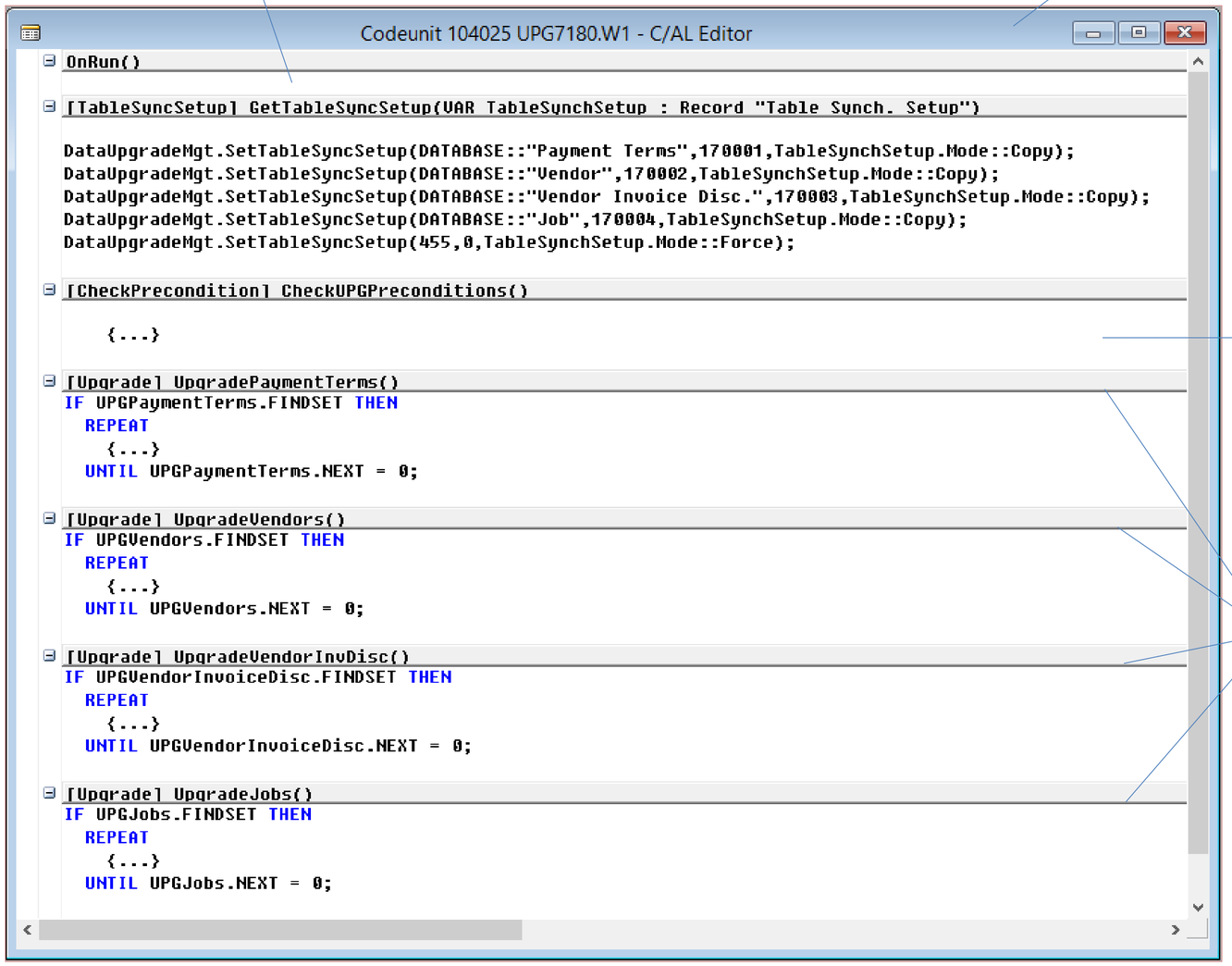
ROOM 9

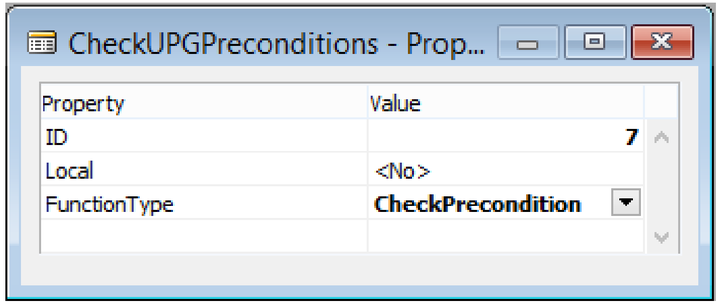
Data Conversion

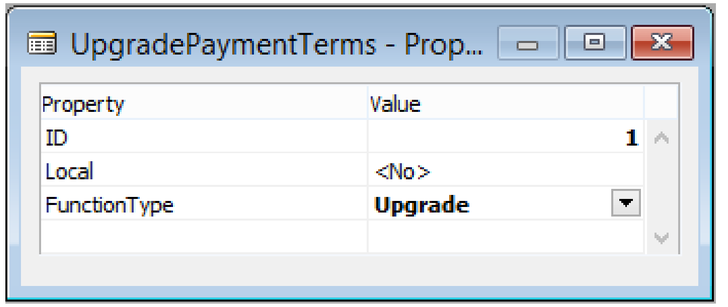
Upgrade Codeunit











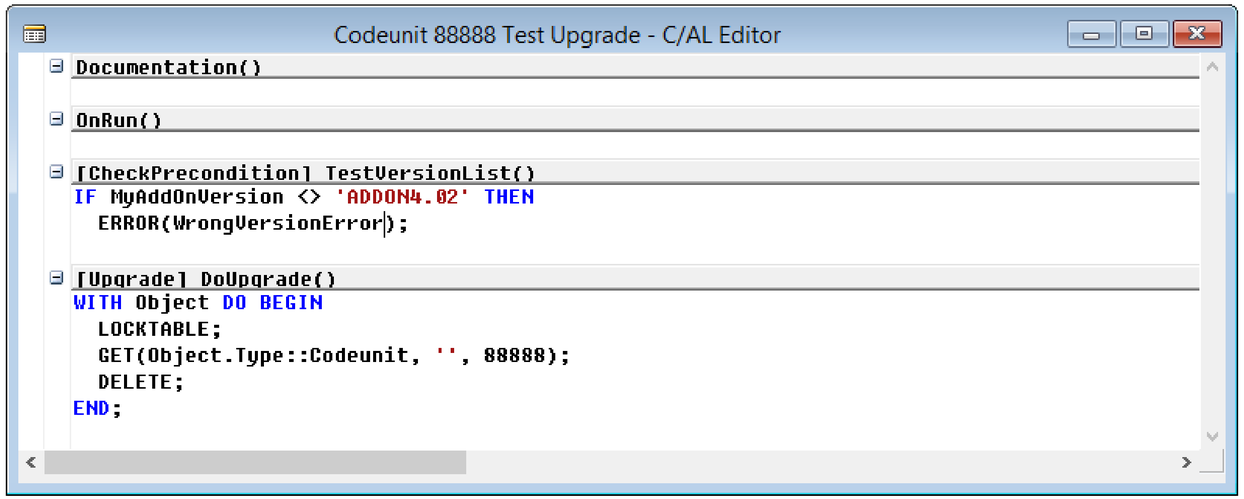
UPGRADE PROCESS

DATA CONVERSION WITH ANY FOB

ANY RELEASE

TEST VERSION

SELF DESTRUCT



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

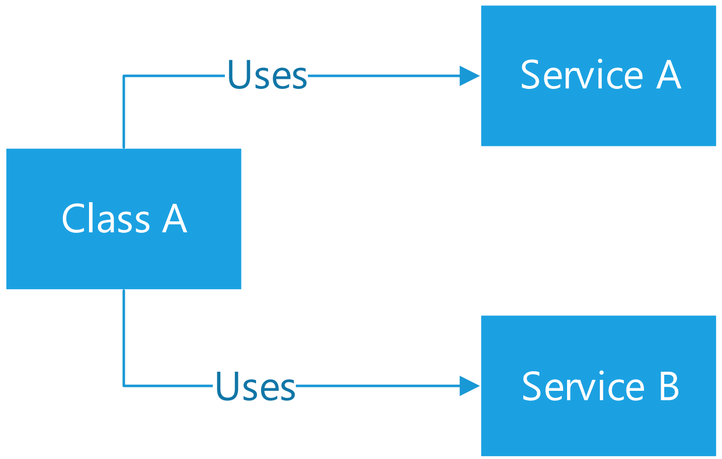
C/AL is very inflexible, not object oriented, and only allows for a few basic patterns (hooks, facade)

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

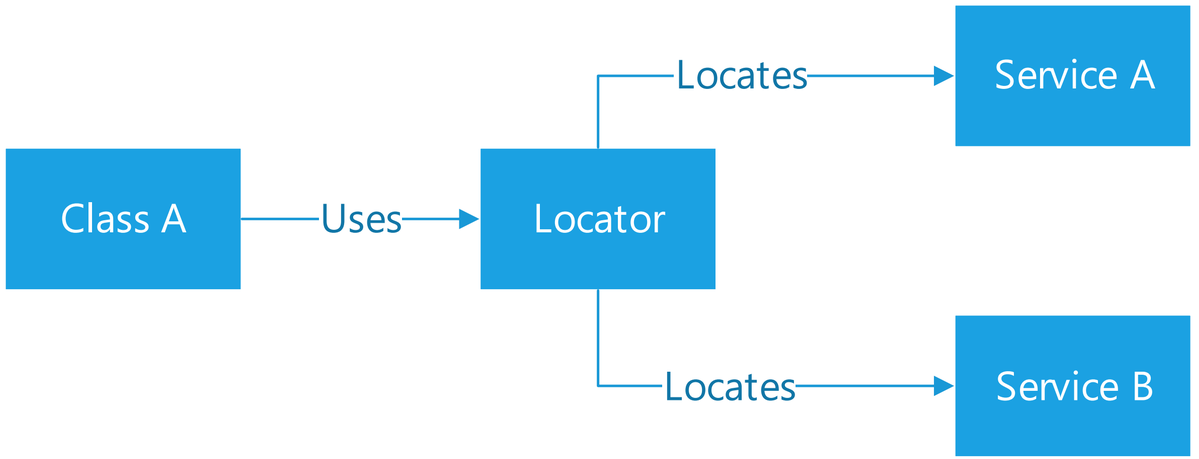
THE SITUATION



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	

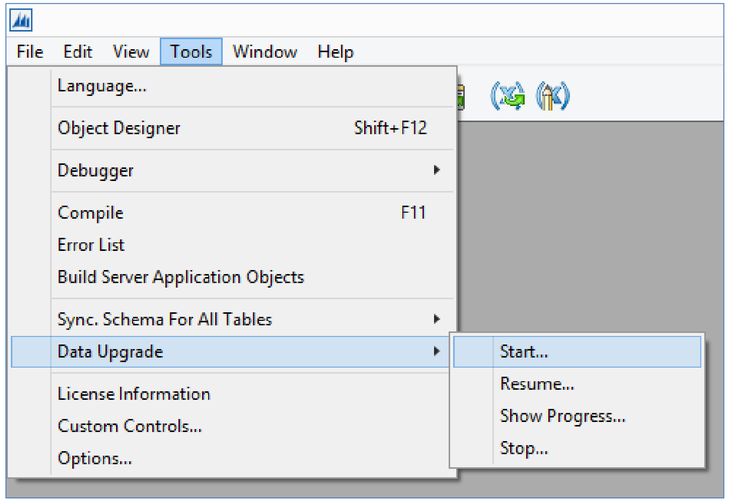
Team Foundation Server

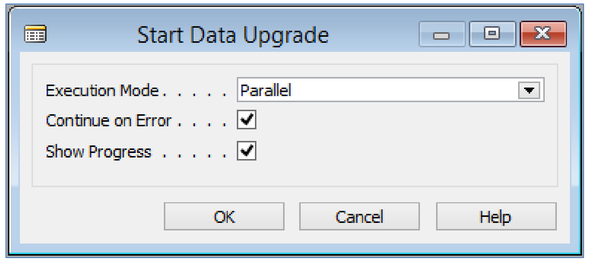
Start-NAVDataUpgrade

Get-NAVDataUpgrade

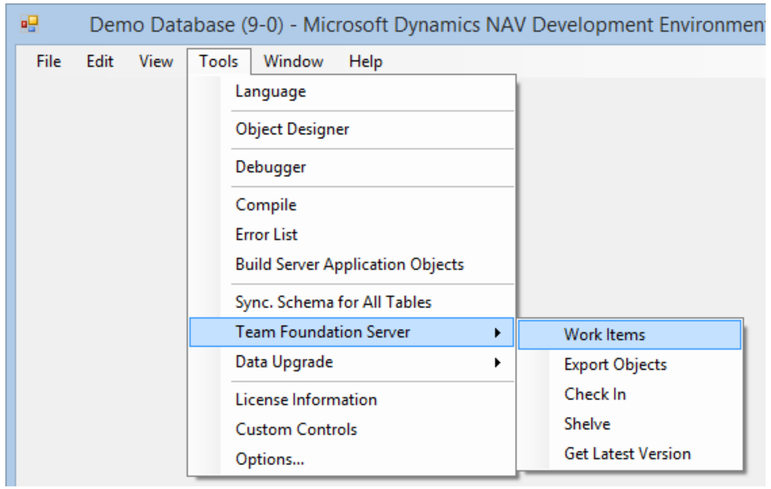
Resume-NAVDataUpgrade

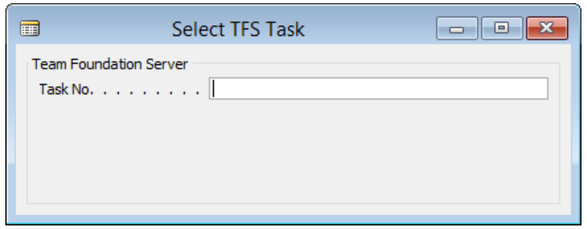
Stop-NAVDataUpgrade





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





CLOUD READY DEVELOPMENT METHODOLOGY WRITING REPEATABLE SOFTWARE

Day 1 – Design Patterns for Repeatable Software

Day 2 – RapidStart Code – PowerShell merging

Day 3 – RapidStart Upgrade – Automated Upgrades

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

Germany

The Netherlands

Belgium

India - New Delhi

Japan - Tokyo

USA - Atlanta

Canada - Toronto

“A Professional Partner Channel Deserves a Professional Work Environment”

Partner Ready Software

It’s YOU!

PRS Partner Ready Software

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.