blueprism

Software Robots - the Virtual Workforce

Blue Prism Learning

OBJECT DESIGN GUIDE

VERSION: 1.0.1

For more information please contact:

info@blueprism.com | UK: +44 (0) 870 879 3000 | US: +1 888 757 7476

www.blueprism.com

Contents

1. Introduction

This guide will outline how to ensure Business Objects are designed to be efficient, scalable and re-useable. The guide will compare a single object per application design against a multi-object per application design and highlight the advantages and disadvantages of each approach.

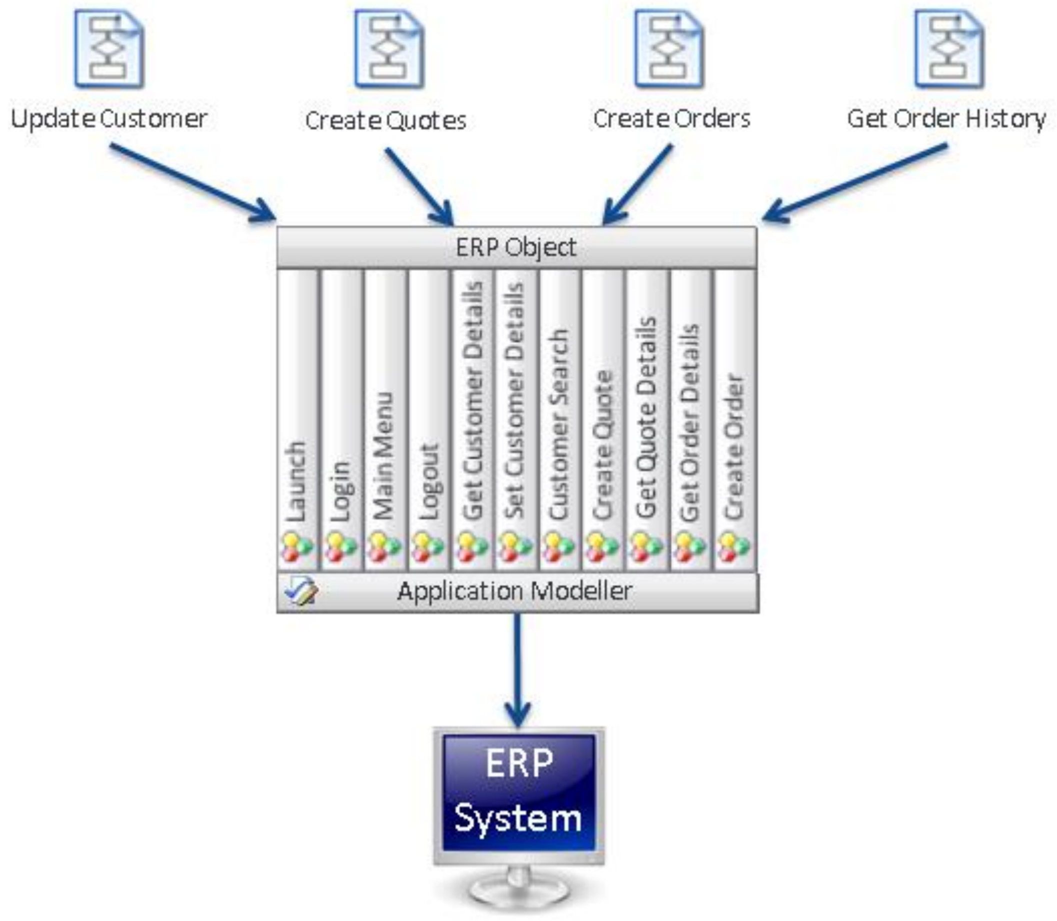
The guide is aimed at Blue Prism Developers and solution architects who have completed the Blue Prism Foundation Course.

After reading this guide you should be able to:

- Descibe the advantages and disadvantages of single-object and multi-object designs
- Design an efficiennt, scalable and re-useable object layer for any application.

2. Single-Object Designs

By single object design we mean that one object is built for an entire application. An example of this is illustrated below:



In the diagram above, we have

- A business application, the ERP System
- A single object, ERP Object, to automate the ERP System
- Four processes, Update Customer, Create Quotes, Create Orders and Get Order History, which call the ERP Object to perform automated tasks against the ERP System.

Whilst this design will work, it provides some challenges and risks:

- Only 1 developer can work on the ERP Object at a time.
- Any process which automates the ERP System must consume into memory all the actions within ERP Object, including the actions it doesn’t use.
- Any change to the ERP Object is effectively a change to any process that calls it
- The REP Object will be larger and less efficient than necessary.

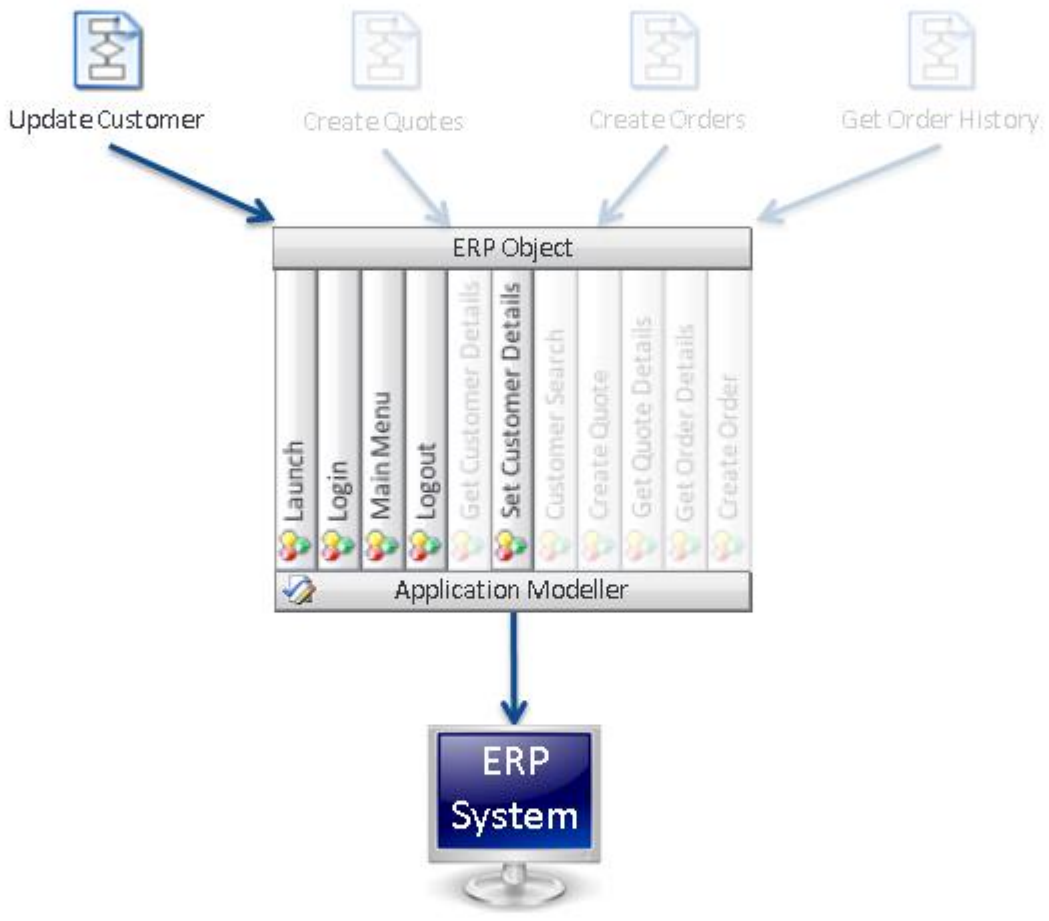
2.1. Only 1 developer can work on the ERP Object at a time

Blue Prism only allows one developer to work on an object at the same time. With the single-object design above, this means that only one developer can be working on the whole of the ERP system interface at a time. Where there is a need for multiple developers to build actions to automate different areas of the ERP System, the single-object design will slow down the development phase.

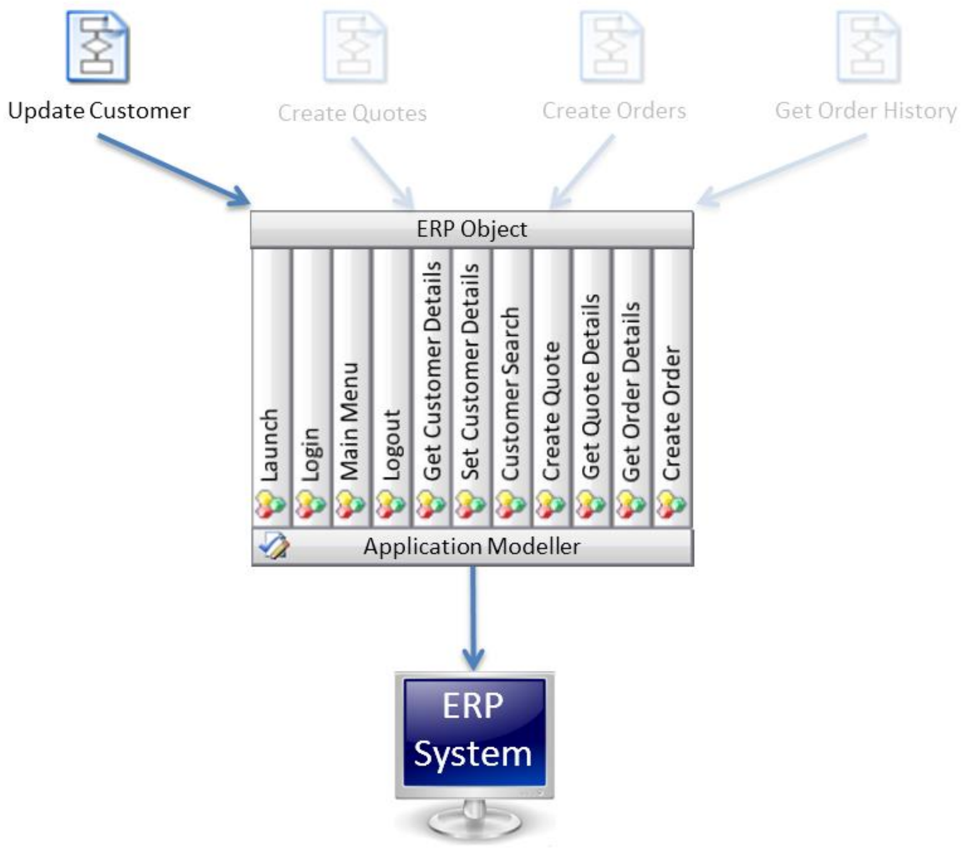
2.2. Any process which automates the ERP System must consume into memory all the actions within ERP Object, including the actions it doesn't use.

The diagram below illustrates how the Update Customer process only uses 5 actions namely

- Launch
- Login
- Main Menu
- Logout
- Set Customer Details

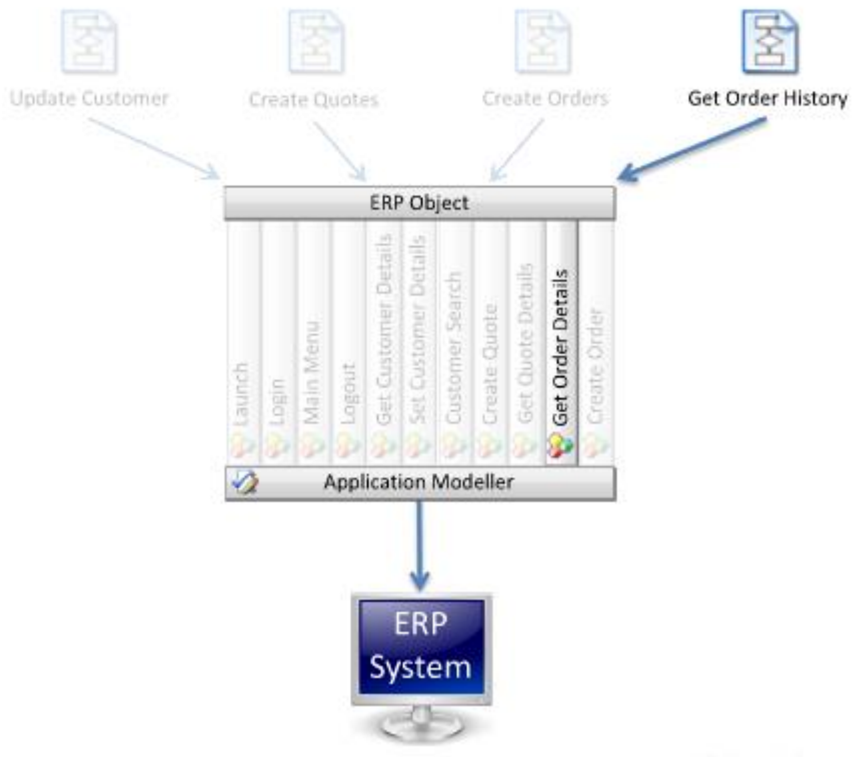


However, with the single-object design, all the actions are consumed into memory when the Update Customer process runs as the entire object is consumed.

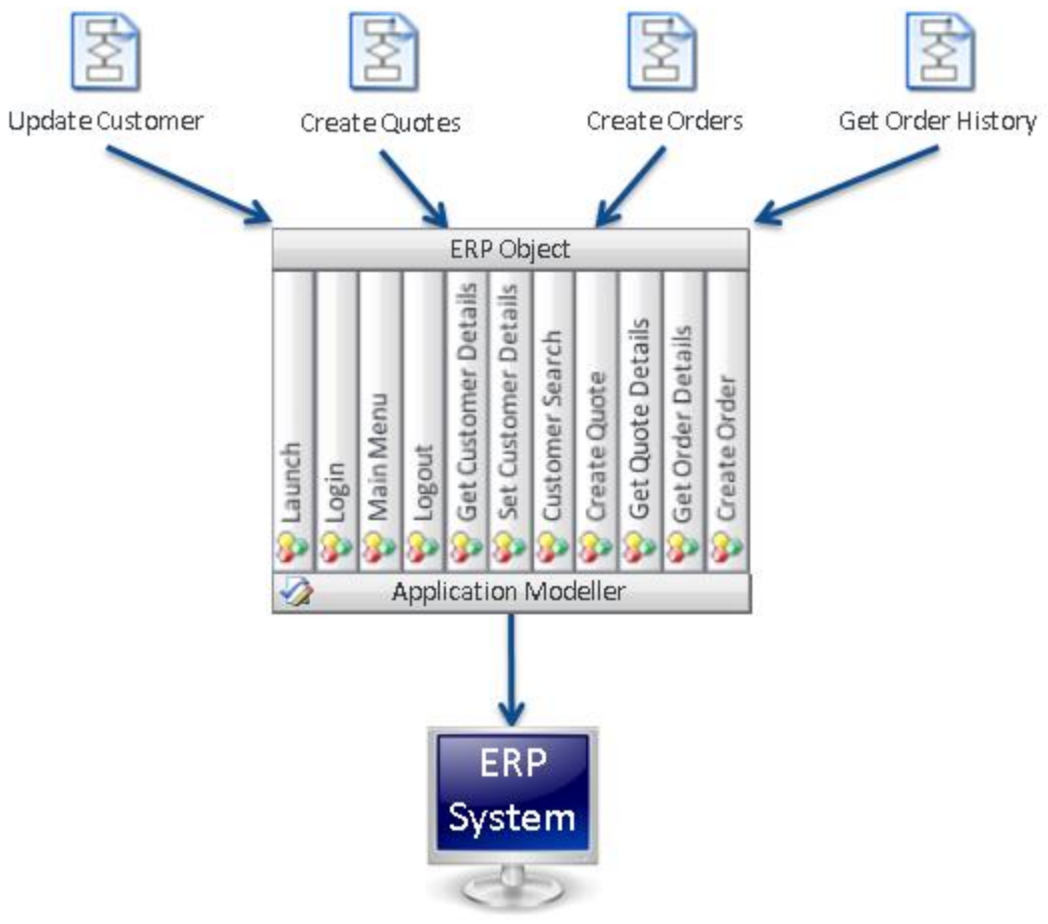


2.3. Any change to the ERP Object is effectively a change to any process that calls it

Suppose the Get Order History process requires a change to be made to the Get Order Details action. The Get Order Details action is only used in the Get Order History process.



However, with the single-object design, a change to the Get Order Details action impacts any process calling the ERP Object. It exposes a risk of latent errors that could affect processes that don't need to be changed. Any change to the ERP Object is effectively a change to any process that uses it.



Although this risk can be mitigated through regression testing, the bigger the ERP Object grows and the more processes that use it, the more regression testing is required for even the smallest of changes to the ERP object.

2.4. The ERP object will be larger and less efficient that necessary

As a single-object, the ERP object will contain all the required elements in the application model and all the actions required to automate the ERP system. This will result in a large file being held on the Blue Prism database. Section 2.2 has described how this large object will be consumed into the memory of any resource pc that is running any process that automates the ERP system, but there is also an impact on the Blue Prism database size.

Each time the ERP object is changed, even if only a single action, a new version of the large object is saved to the database with the older version being kept in the object history. The performance of the interactive client machine when developing against a large object will also be adversely impacted.

As a single-object, the ERP object will have a large Application Model, which can be difficult to navigate and pose an increased risk of the wrong element being changed.

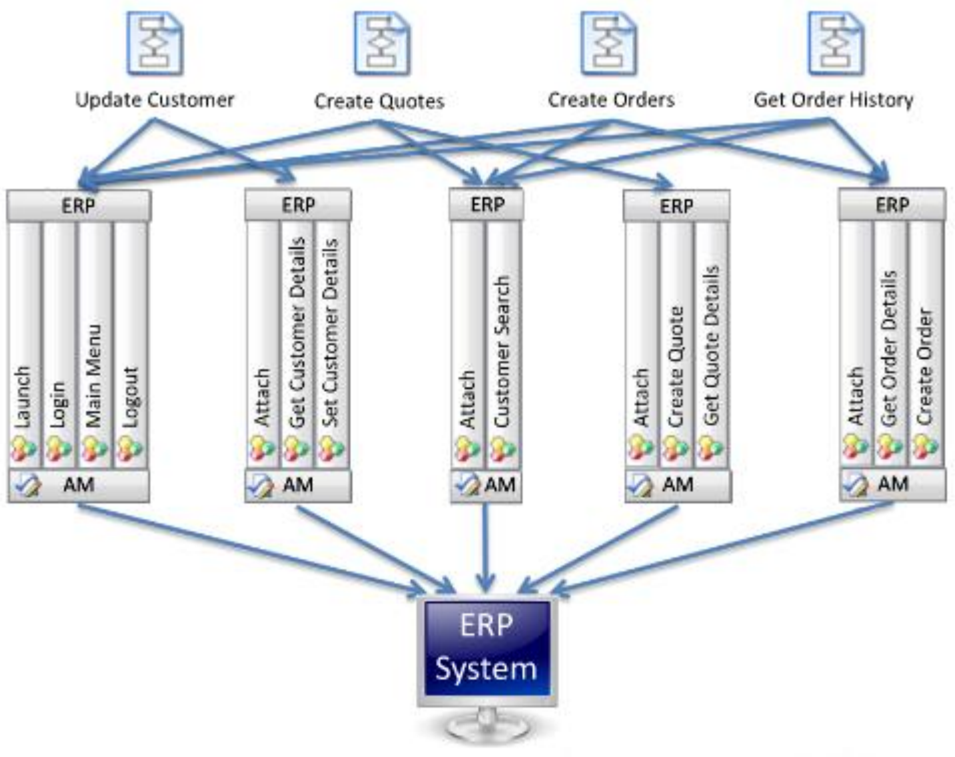
2.5. When is a Single-Object Design appropriate

A single-object design is appropriate for a small proof of technology exercise or a proof of concept project where delivery is done by a single developer ***and the object created will not subsequently be promoted to a production environment.***

In all other scenarios, a more efficient and scalable design is required using a multi-object design.

3. Multi-Object Designs

By multi-object design we mean that more than one object is built for an application. Although there are no hard and fast rules regarding how many objects are built, a good rule of thumb to work by is to build a single object for each screen that is to be automated. An example of a multi-object design is illustrated below:



In the diagram above, we now have

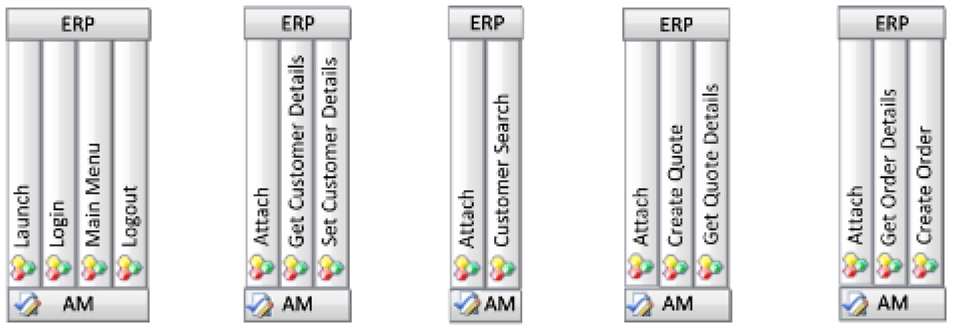
- A business application, the ERP System
- Five business objects to automate the ERP System
- Four processes, Update Customer, Create Quotes, Create Orders and Get Order History, which call the required objects to perform the relevant automated tasks against the ERP System

This design is more scalable and efficient because:

- 5 developers can now build objects that automate the ERP System at the same time
- Any process which automates the ERP System only consumes less actions into memory

- A change to the actions within the ERP object will impact less processes
- The individual objects will be smaller and more efficient with a smaller Application Model

3.1. 5 developers can now build objects that automate the ERP System at the same time

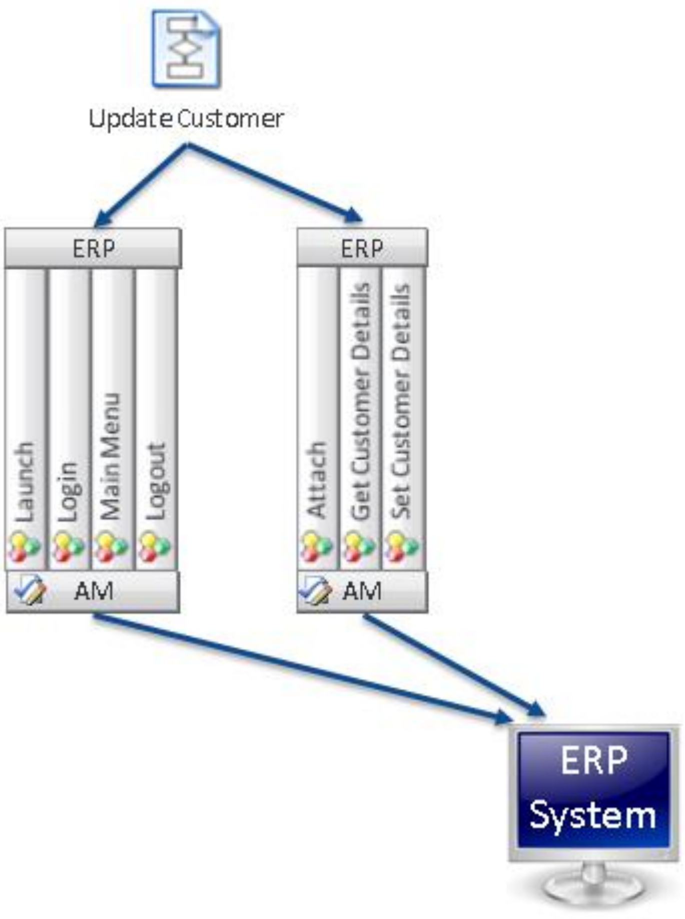


There are now five separate business objects that automate different parts of the ERP system. This enables up to five developers to work on objects automating the ERP system at any time. Over time, as more areas of the ERP system are automated, the multi-object approach will mean additional objects are built thus enabling more developers to work on ERP system Object. The multi-object approach is more scalable and facilitates faster development times.

3.2. Any process which automates the ERP System only consumes less actions into memory

Section 2.2 discussed how the Update Customer process only uses 5 actions namely

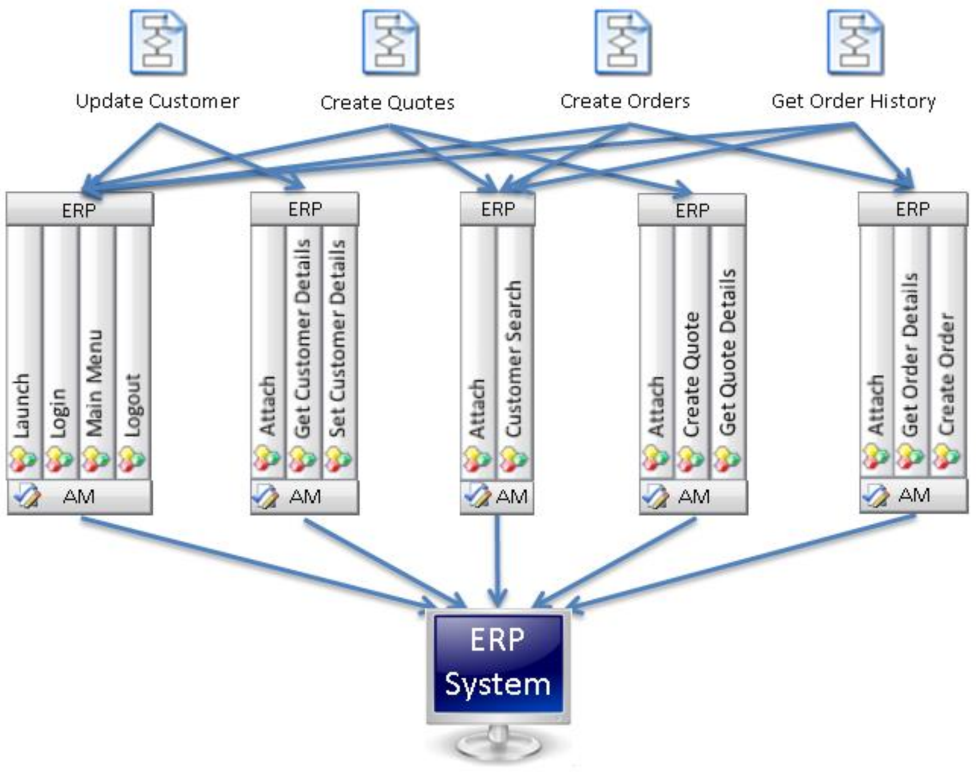
- Launch
- Login
- Main Menu
- Logout
- Set Customer Details



With the multi-object design approach, the Update Customer process now consumes much fewer actions into memory at run time.

3.3. A change to the actions within the ERP object will impact less processes

In section 2.3 we looked at how the Get Order History process requires a change to be made to the Get Order Details action. The Get Order Details action is only used in the Get Order History process.



With the multi-object design approach above, a change to the Get Order Details action, will only impact the Get Order History and Create Order processes as these are the only processes calling the object that contains the Get Order Details action. As a result of adopting the multi-object approach, the amount of regression testing required has been reduced by 50%.

3.4. The individual objects will be smaller and more efficient with a smaller application model

Each individual contains less actions and a smaller application model as only the elements required for the set of actions contained within the objects need to be defined. Subsequently, the individual objects consume less space on the Blue Prism database, the database size grows less when a single action is changed and the application model is more user friendly with less of a risk of the wrong element being amended.

3.5. When is a multi-object design appropriate

All projects where it is possible that the business objects will end up in production, regardless of the size of the initial development team.

Any proof of concept project whereby multiple developers are required to develop against an individual application.

4. Multi-Object Design Example

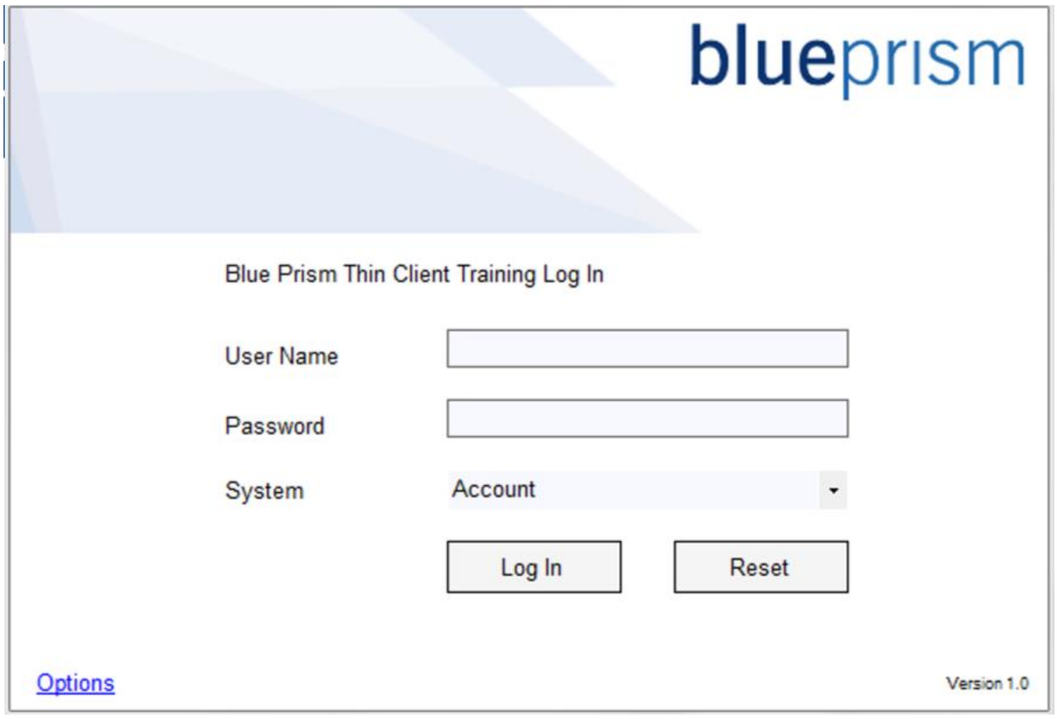
The following example illustrates how to design an efficient, scalable and re-useable object layer.

4.1. Basic Actions - Example

All applications will require a ‘Basic Actions’ Object. This will contain actions for tasks such as launching the application logging in, closing the application and any other actions that not screen-specific such as ‘Go Home’. In our example we have the following actions defined in our ‘Basic Actions’ Object.

Action	Inputs	Outputs
Launch		
Login	Username, Password, System	
Terminate		

The only screen that this object is automating is the Login Screen.



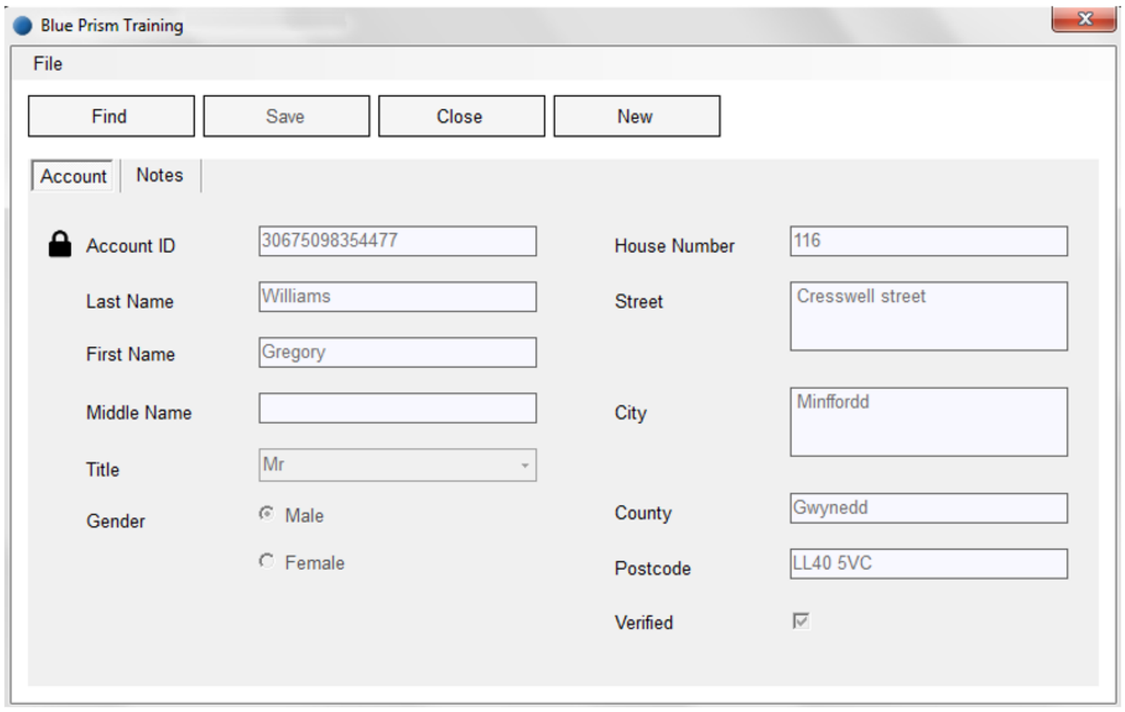
Note how all the possible fields to be completed are driven by Inputs. Even if the first Blue Prism process which calls the Object uses the same value for a field, populating that field should still always be driven by an input parameter to facilitate future re-use.

4.2. Other Objects - Examples

As discussed in section 3, other objects will be typically at an object-per-screen level. Below are some more examples. Note how all the actions that write or set data are ‘told’ by the calling process what data to write in the form of inputs. There is no need to have a separate action for each data item to be written, all fields can be populated within a single action.

Actions that read or get data should simply read the data from the screen and pass it back to the calling process. The calling process can then apply any process specific business logic to the data. By following this approach and keeping any business logic out of the object layer enables maximum re-use of the object layer. There is no need to have a separate action for each data item to be read, all fields can be read within a single action.

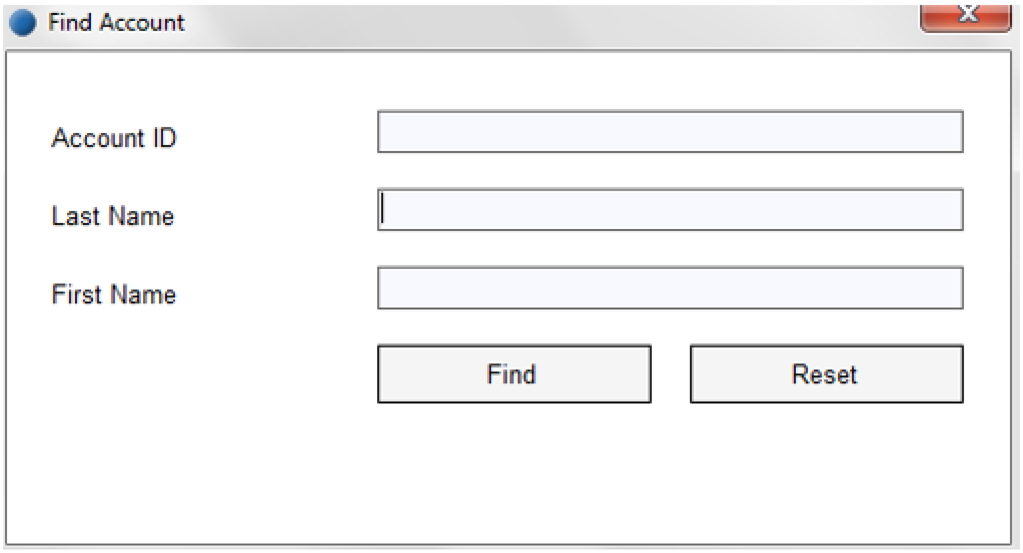
4.2.1 Account Details Screen

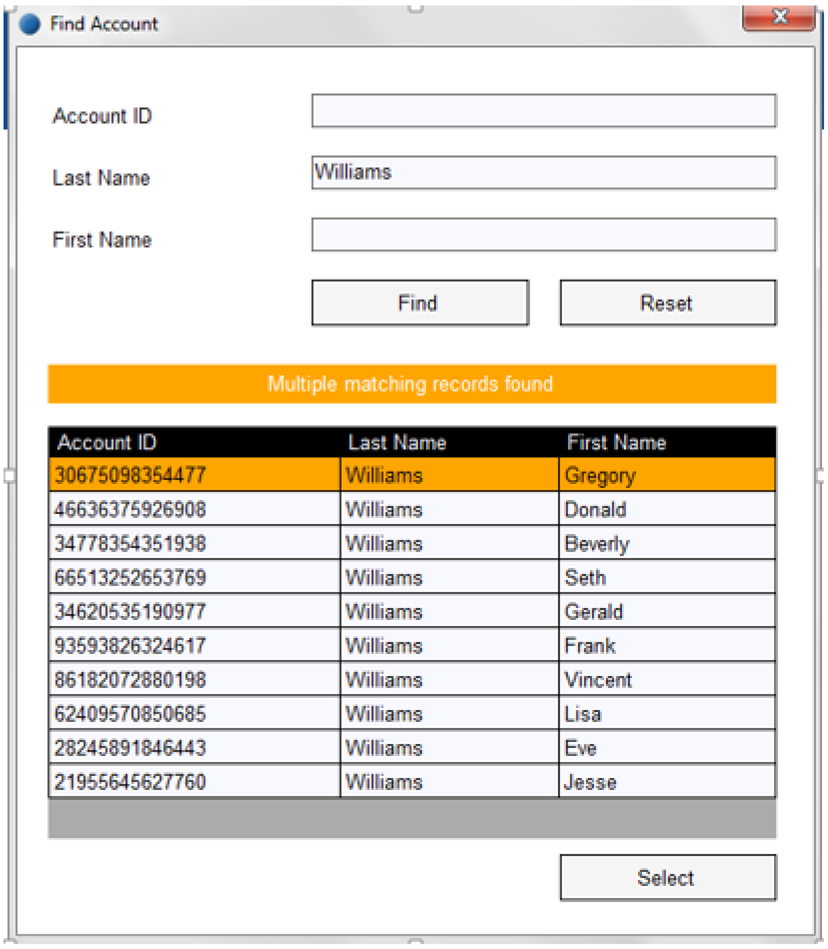


Account ID	30675098354477	House Number	116
Last Name	Williams	Street	Cresswell street
First Name	Gregory		
Middle Name		City	Minffordd
Title	Mr		
Gender	(●) Male ( ) Female	County	Gwynedd
		Postcode	LL40 5VC
		Verified	✓

Action	Inputs	Outputs
Attach		
Navigate	Find, Save, Close, New, Lock/Unlock	
Get Account Details		Account ID, Last Name, Middle Name, First Name, Title, Gender, House No., Street, City, County, Postcode, Verified.
Set Account Details	Account ID, Last Name, Middle Name, First Name, Title, Gender, House No., Street, City, County, Postcode, Verified.	

4.2.2 Find Account





Account ID	
Last Name	Williams
First Name	

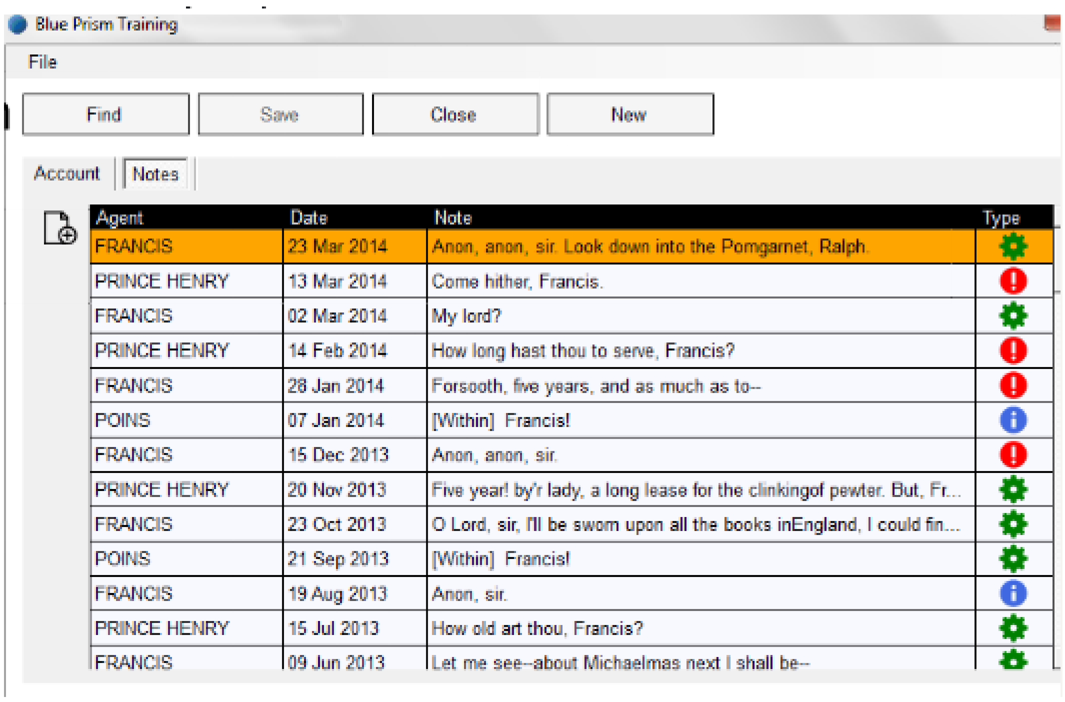
Multiple matching records found

Account ID	Last Name	First Name
30675098354477	Williams	Gregory
46636375926908	Williams	Donald
34778354351938	Williams	Beverly
66513252653769	Williams	Seth
34620535190977	Williams	Gerald
93593826324617	Williams	Frank
86182072880198	Williams	Vincent
62409570850685	Williams	Lisa
28245891846443	Williams	Eve
21955645627760	Williams	Jesse

Action	Inputs	Outputs
Attach		
Find Account	Account ID, Last Name, First Name	Found

In this example, the Find Account action will perform a search based on the inputs provided and return to the calling process the results in a collection called Found.

4.2.3 Notes



Agent	Date	Note	Type
FRANCIS	23 Mar 2014	Anon, anon, sir. Look down into the Pomgarnet, Ralph.	⚙
PRINCE HENRY	13 Mar 2014	Come hither, Francis.	❗
FRANCIS	02 Mar 2014	My lord?	⚙
PRINCE HENRY	14 Feb 2014	How long hast thou to serve, Francis?	❗
FRANCIS	28 Jan 2014	Forsooth, five years, and as much as to--	❗
POINS	07 Jan 2014	[Within] Francis!	ℹ
FRANCIS	15 Dec 2013	Anon, anon, sir.	❗
PRINCE HENRY	20 Nov 2013	Five year! by'r lady, a long lease for the clinkingof pewter. But, Fr...	⚙
FRANCIS	23 Oct 2013	O Lord, sir, I'll be sworn upon all the books inEngland, I could fin...	⚙
POINS	21 Sep 2013	[Within] Francis!	⚙
FRANCIS	19 Aug 2013	Anon, sir.	ℹ
PRINCE HENRY	15 Jul 2013	How old art thou, Francis?	⚙
FRANCIS	09 Jun 2013	Let me see--about Michaelmas next I shall be--	⚙

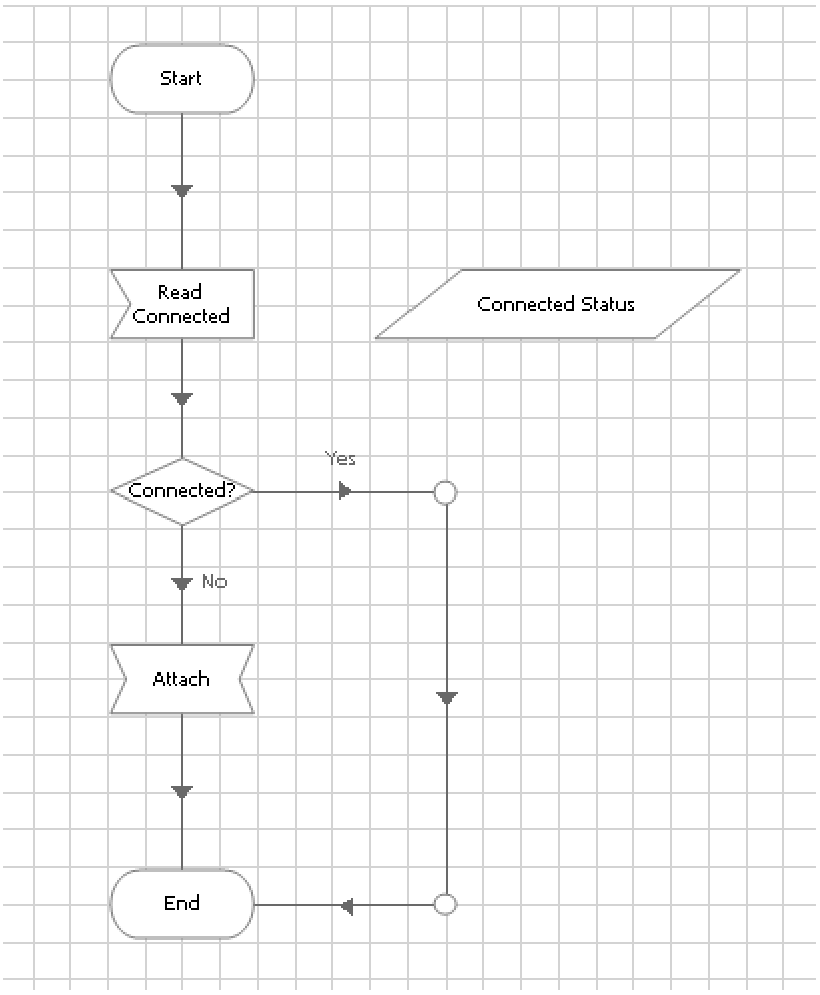
Action	Inputs	Outputs
Attach		
Get Notes		Collection(Agent, Date, Note, Type)

4.2.4 Attaching

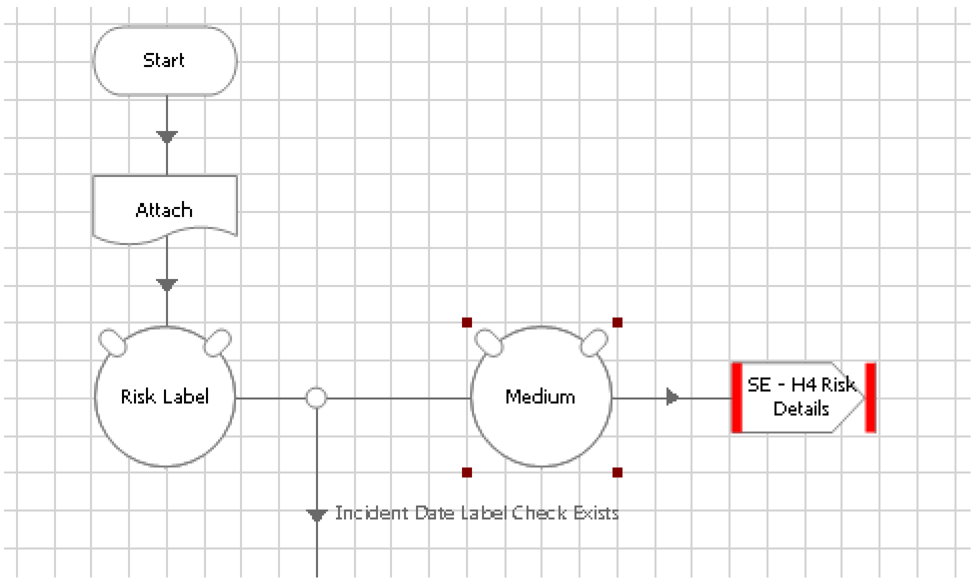
You may have noticed that all Objects except for 'Basic Actions' have an action defined called 'Attach'. An object must be attached to the application before it can be used to automate it. When an object launches an application, it is automatically attached to that application. Therefore, the 'Basic Actions' object does not require an 'Attach' action. The remaining objects that wish to work with an application that is already launch must first attach to the application.

4.2.1.1. Attaching Best Practice

If an object attempts to attach to an application when it is already attached, an error will result. Therefore, when building an 'Attach' action, it is best practice to first detect if the object is already attached to the application. A typical 'Attach' action may look like this



By using the approach above, every other action within the object can call the 'Attach' page as is its first stage to ensure the action is ready to work with the application e.g.



5. Naming Conventions

Avoid using terms that are process specific in your objects and actions. Remember an object is an application interface which should be completely independent of any process which may use it.

Objects names should be kept to ***{Application Name – Screen Name}*** format for example, PeopleSoft – Employee Details.

Action names should be kept generic and provide an explanation of what the action does for example Write Data, Read Data, Navigate to Salary Details Screen.

Using a combination of the above enables future users of the object to very quickly understand what tasks individual actions perform.

6. Object Design – 5 Golden Rules

- Use a multi-object design approach
- Keep actions small and limitied to a single specific task (e.g. read, write, navigate)
- Do not include process specific business logic in an object
- Use input parameters to drive what data is entered into an application and determine the contents of these parameters in the process layer
- Use output parameters to pass back values of fields held on the application. Where a process requires business logic to be applied to data held on the application, apply that logic in the process layer.