Практикум управления требованиями. Пример 1. Автоматизация заполнения атрибутов.

Автор: **Дмитрий Зиновьев**
старший руководитель проектов Citia BTC

В статье **Управление большими объемами требований в BI-проектах** я упоминал несколько возможных применений VBS-скриптов в целях автоматизации и увеличения эффективности работы бизнес-аналитика. В обзорном материале я намеренно не вдавался в детали таких скриптов, однако, как мне кажется было бы полезным сделать более подробный разбор таких случаев.

Наиболее востребованный пример, когда при работе с PowerDesigner требуется автоматизация работы – это распространение значений атрибутов внутри иерархии определенного типа требований или по связанным требованиям.

Предположим, что у нас в **модели** имеется два типа требований: аналитические задачи (стереотип Analytical task) и показатели (Indicator). Аналитические задачи выстроены в иерархии по подразделениям (стереотип Task group), а показатели – по видам анализа (стереотип Subject area). При этом, для каждой задачи через связь (Traceability link) указаны те показатели, которые требуются для решения задач.

		1. Продажи				2. Расходы		3. Продукты	
		1.1 Количество продаж фактическое	1.2 Сумма продажи фактическая	1.3 Количество продаж прогнозное	1.4 Сумма продаж прогнозная	2.1 Стоимость продвижения продукта	2.2 Затраты на продажи	3.1 Количество продаж продукта	3.2 Сумма проданных товаров
1. Задачи дирекции по продажам	1.1 Анализ фактического объема продаж	X	X						
	1.2 Прогноз продаж			X	X				
2. Задачи финансовой	2.1 Анализ доходов		X		X				X
	2.2 Анализ расходов					X	X		
3. Задачи дирекции по маркетингу	3.1 Анализ затрат на продвижение продуктов					X			
	3.2 Анализ эффективности продвижения продуктов	X	X			X		X	X

Рисунок 1. Зависимость задач от показателей

В модели задачи и показатели разнесены в разные пакеты.

Реализация всех задач планируется в три основных этапа, на каждом из которых будут решены задачи определенных подразделений, данная информация внесена в пользовательский атрибут «Stage» на верхнем уровне иерархии аналитических задач. Требуется проставить этап реализации для всех задач, а также показателей. Этапом реализации показателя должен стать наиболее ранний этап из тех задач, для которых нужен этот показатель. Предполагается, что состав подразделений в этапах может многократно пересматриваться, поэтому требуется актуализация этапов реализации задач и показателей.

При детальном рассмотрении данной ситуации становится понятно, что ручная проработка всех задач и связанных с ними показателей превратит труд аналитика в бесконечное проставление значений атрибутов вместо собственно анализа и обоснования наполнения этапов реализации. Для того, что бы избежать ручной проработки всех задач в иерархии и связанных с ними показателей необходимо создать обработчик события для стереотипа Task group, который при изменении этапа реализации будет автоматически проставлять это значение и, при необходимости, транслировать его в показатель.

Для того, что бы создать обработчик событий, необходимо зайти в расширения модели (local extensions), найти необходимый стереотип (Task group) для объекта требований и создать в нем обработчик. Поскольку нам необходимо проверять этапы реализации при каждом изменении, необходимо выбрать тип события – Validate, которое срабатывает при применении изменений в объекте требований искомого стереотипа.

Прежде всего, нам необходимо проставить этап в задачи группы, на которой обрабатывается изменение. Для этого необходимо перебрать все объекты в коллекции Children, которая содержит дочерние объекты требований для обрабатываемого требования. Для найденной задачи необходимо сделать три вещи: проставить этап реализации, проверить наличие дочерних задач и проверить имеющиеся связи.

Установить значение расширенного атрибута нужно с помощью метода SetExtendedAttribute, а получить значение установленного этапа с помощью метода GetExtendedAttribute, при этом в параметре метода следует указать название атрибута Stage.

Для того, что бы проверить связи обрабатываемого объекта необходимо перебрать коллекцию OutgoingTraceabilityLinks, которая содержит все объекты связи, отражающие зависимость обрабатываемого требования от других объектов. К объекту-источнику связи можно обратиться через свойство LinkedObject в объекте связи. Прежде чем обращаться к атрибутам, не лишим будет убедиться в том, что потомок относиться к классу требований и является показателем.

Поскольку задачи и показатели у нас разнесены в разные пакеты, то связи между ними автоматически генерируются через объекты ссылки (Shortcut), поэтому нам следует обработать и их. В Shortcut хранится ссылка на исходный объект в свойстве TargetObject

```
dim Trace, Traced_obj

for each Trace in Task.OutgoingTraceabilityLinks

    if (Trace.LinkedObject.isKindOf(cls_Requirement))then

      if (Trace.LinkedObject.Stereotype = "Indicator") then
```

```
        Set Traced_obj = Trace.LinkedObject
      end if
    else
      if (Trace.LinkedObject.isKindOf(cls_Shortcut)) then
        if (Trace.LinkedObject.TargetObject.isKindOf(cls_Requirement))then
          if (Trace.LinkedObject.TargetObject.Stereotype = "Indicator") then
            Set Traced_obj = Trace.LinkedObject.TargetObject
          end if
        end if
      end if
    end if
  next
```

Что бы произвести аналогичные действия для всех потомков исходной группы задач оформим обработку связей в виде рекурсивной функции.

```
sub Check_Traces(Task)
  dim Trace, Min_Stage, Traced_obj, Task_Stage

  Task_Stage = Task.GetExtendedAttribute("Stage")
  set Traced_obj = nothing
  Min_Stage = 0

  for each Trace in Task.OutgoingTraceabilityLinks
    if (Trace.LinkedObject.isKindOf(cls_Requirement))then
      if (Trace.LinkedObject.Stereotype = "Indicator") then
        Set Traced_obj = Trace.LinkedObject
      end if
    else
      if (Trace.LinkedObject.isKindOf(cls_Shortcut)) then
        if (Trace.LinkedObject.TargetObject.isKindOf(cls_Requirement))then
```

```
                if (Trace.LinkedObject.TargetObject.Stereotype = "Indicator") then
                  Set Traced_obj = Trace.LinkedObject.TargetObject
                end if
              end if
            end if
          end if
          if (not (Traced_obj is nothing)) then
            Min_Stage = Find_Earliest_Link(Traced_obj)
            if (Min_Stage > 0) then
              if (Task_Stage < Min_Stage) then
                Traced_obj.SetExtendedAttribute "Stage", Task_Stage
              else
                Traced_obj.SetExtendedAttribute "Stage", Min_Stage
              end if
            end if
            Check_Traces(Traced_obj)
          end if
          Min_Stage = 0
          set Traced_obj = nothing
          set Trace = nothing
        next
end sub
```

Следует принимать во внимание, что каждый показатель может быть связан с несколькими задачами, поэтому для каждого показателя следует проверить те задачи, в которых он рассчитывается, что бы определить минимальный этап, на котором этот показатель потребуется. Перечень объектов, которые зависят от данного экземпляра, хранится в коллекции IncomingTraceabilityLinks. Создадим функцию, возвращающую минимальный этап:

```
      function Find_Earliest_Link(Indicator)
        dim Trace, Earliest, Trace_Stage
        Earliest = 0
        Trace_Stage = 0
```

```
        for Each Trace in Indicator.IncomingTraceabilityLinks
          if (Trace.SourceObject.IsKindOf(cls_Requirement)) then
            if (not Trace.SourceObject.GetExtendedAttributeText("Stage") = "") then
              Trace_Stage = Trace.SourceObject.GetExtendedAttribute("Stage")
            end if
          else
            if (Trace.SourceObject.IsKindOf(cls_Shortcut)) then
              if Trace.SourceObject.TargetObject.IsKindOf(cls_Requirement) then
                if (not Trace.SourceObject.TargetObject.GetExtendedAttributeText("Stage") = "") then
                  Trace_Stage = Trace.SourceObject.TargetObject.GetExtendedAttribute("Stage")
                end if
              end if
            end if
          end if
          if (Trace_Stage > 0) then
            if (Earliest = 0) or (Earliest > Trace_Stage) then
              Earliest = Trace_Stage
            end if
          end if
          Trace_Stage = 0
        next
        Find_Earliest_Link = Earliest
end function
```

Для того, что бы сделать результаты обработки видимой для пользователя добавим вывод описания производимых действий и их результатов в окно Results. Ну и, конечно же, не стоит забывать очищать созданные объекты после завершения работы обработчика. Таким образом, готовый скрипт имеет следующий вид:

```
Function %Validate%(obj, ByRef message)
```

```
  if (obj.GetExtendedAttribute("Stage") > 0) then
    output "Проверяем дочерние задачи группы " + obj.Name
    Check_Task_Children(obj)
  end if

  %Validate% = True

End Function
sub Check_Task_Children(Task_group)
  dim Task
  for each Task in Task_group.Children
    if (Task.IsKindOf(cls_Requirement)) and (Task.Stereotype = "Analytical task") then
      output "Найдена дочерняя задача " + Task.DisplayName
      Task.SetExtendedAttribute "Stage", Task_group.GetExtendedAttribute("Stage")
      output "Задаче установлен этап " + Task.GetExtendedAttribute("Stage")
      output "Проверяем связи задачи " + Task.DisplayName
      Check_Traces(Task)
      output "Проверяем дочерние задачи для задачи " + Task.DisplayName
      end if
      output ""
  next
set Task = nothing
end sub
sub Check_Traces(Task)
  dim Trace, Min_Stage, Traced_obj, Task_Stage

  Task_Stage = Task.GetExtendedAttribute("Stage")
  output "На входном требовании этап "
  output Task_Stage
```

```
set Traced_obj = nothing
Min_Stage = 0

for each Trace in Task.OutgoingTraceabilityLinks
  if (Trace.LinkedObject.isKindOf(cls_Requirement))then
    if (Trace.LinkedObject.Stereotype = "Indicator") then
      Set Traced_obj = Trace.LinkedObject
    end if
  else
    if (Trace.LinkedObject.isKindOf(cls_Shortcut)) then
      if (Trace.LinkedObject.TargetObject.isKindOf(cls_Requirement))then
        if (Trace.LinkedObject.TargetObject.Stereotype = "Indicator") then
          Set Traced_obj = Trace.LinkedObject.TargetObject
        end if
      end if
    end if
  end if
  if (not (Traced_obj is nothing)) then
      output " В связях найден показатель " + Traced_Obj.DisplayName + ", на показателе сейчас этап " +
Traced_Obj.GetExtendedAttribute("Stage")

    Min_Stage = Find_Earliest_Link(Traced_obj)

    output "  Итого минимальный этап по связям сейчас "
    output Min_Stage

    if (Min_Stage > 0) then
      if (Task_Stage < Min_Stage) then
        Traced_obj.SetExtendedAttribute "Stage", Task_Stage
```

```
            else
              Traced_obj.SetExtendedAttribute "Stage", Min_Stage
            end if
          end if
          output "Проверяем связи найденного показателя"
          Check_Traces(Traced_obj)
        end if
        output ""
        Min_Stage = 0
        set Traced_obj = nothing
        set Trace = nothing
      next
end sub
function Find_Earliest_Link(Indicator)
  dim Trace, Earliest, Trace_Stage
  Earliest = 0
  Trace_Stage = 0
  output "    Ищем входящие связи для определения минимального этапа"
  for Each Trace in Indicator.IncomingTraceabilityLinks
    if (Trace.SourceObject.IsKindOf(cls_Requirement)) then
        output "      Найдена входящая связь на " + Trace.SourceObject.DisplayName + " там этап " + Trace.SourceObject.GetExtendedAttribute("Stage")
      if (not Trace.SourceObject.GetExtendedAttributeText("Stage") = "") then
        Trace_Stage = Trace.SourceObject.GetExtendedAttribute("Stage")
      end if
    else
      if (Trace.SourceObject.IsKindOf(cls_Shortcut)) then
        if Trace.SourceObject.TargetObject.IsKindOf(cls_Requirement) then
          output "      Найдена входящая связь на " + Trace.SourceObject.TargetObject.DisplayName + " там этап " + Trace.SourceObject.TargetObject.GetExtendedAttribute("Stage")
```

```
                    if (not Trace.SourceObject.TargetObject.GetExtendedAttributeText("Stage") = "") then
                      Trace_Stage = Trace.SourceObject.TargetObject.GetExtendedAttribute("Stage")
                    end if
                  end if
                end if
              end if
              if (Trace_Stage > 0) then
                if (Earliest = 0) or (Earliest > Trace_Stage) then
                  Earliest = Trace_Stage
                end if
              end if
              Trace_Stage = 0
            next
          set Trace = nothing
          Find_Earliest_Link = Earliest
end function
```