$DocumentName

$Author

Revision: $Revisions.lastChild.name

|  |  |
| --- | --- |
|  |  |
| $CompanyName | $date.get(“MMMMM dd, yyyy”) |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Reason For Changes** | **Author** |
| #forrow($rev in $Revisions)$rev.name | $rev.date | $rev.description | $rev.author#endrow |

The latest version of this document may be retrieved at <https://pages.nist.gov/ElectionModeling/>

#if($report.isNull($Diagram))

#set($Diagram = [])

#end

## Legend Support (JND 3/18)

#set($stereotypesInDiagram = $array.createHashSet())

#macro (**removeLineBreak $s**)

#set ($removeLineBreak = “”)

#foreach ($c in $s.toCharArray())#if($c.getType($c)==15)#set($removeLineBreak = $removeLineBreak.concat(“ “))#else#set($removeLineBreak = $removeLineBreak.concat($c.toString()))#end#end

#if ($removeLineBreak == “”)#set($removeLineBreak = $empty)#end

#end

#macro (**escapeCamelCase $s**)

#set ($escapeCamelCase = $s.replaceAll(“[A-Z]”, “ $0”).trim())

#set ($escapeCamelCase = $escapeCamelCase.substring(0, 1).toUpperCase().concat($escapeCamelCase.substring(1)))

#end

#macro (**resolveNotationName $element**)

#set ($resolveNotationName = “”)

#foreach ($stereotype in $element.appliedStereotype)

#if ($stereotype.owner.name == “BPMN2 Profile”)

#set ($name = $stereotype.name)

#escapeCamelCase($name)

#set ($resolveNotationName = $escapeCamelCase)

#end

#if ($resolveNotationName == “”)

#set ($resolveNotationName = $element.humanType)

#end

#end

#end

#macro (**resolveStereotypeName $element**)

#set ($resolveStereotypeName = “”)

#foreach ($stereotype in $element.appliedStereotype)

#if ($resolveStereotypeName == “”)

#set ($name = $stereotype.name)

#escapeCamelCase($name)

#set ($resolveStereotypeName = $escapeCamelCase)

#end

#end

#if ($resolveStereotypeName == “”)

#set ($resolveStereotypeName = $element.humanType)

#end

#end

#set ($acceptedNotation = $array.createArray())

#set ($void = $acceptedNotation.add(“Exclusive Gateway”))

#set ($void = $acceptedNotation.add(“Inclusive Gateway”))

#set ($void = $acceptedNotation.add(“Complex Gateway”))

#set ($void = $acceptedNotation.add(“Event Based Gateway”))

#set ($void = $acceptedNotation.add(“Parallel Gateway”))

#set ($void = $acceptedNotation.add(“Task”))

#set ($void = $acceptedNotation.add(“Service Task”))

#set ($void = $acceptedNotation.add(“Send Task”))

#set ($void = $acceptedNotation.add(“Script Task”))

#set ($void = $acceptedNotation.add(“Receive Task”))

#set ($void = $acceptedNotation.add(“User Task”))

#set ($void = $acceptedNotation.add(“Manual Task”))

#set ($void = $acceptedNotation.add(“Business Rule Task”))

#set ($void = $acceptedNotation.add(“SubProcess”))

#set ($void = $acceptedNotation.add(“Transaction SubProcess”))

#set ($void = $acceptedNotation.add(“AdHoc SubProcess”))

#set ($void = $acceptedNotation.add(“Call Activity”))

#set ($void = $acceptedNotation.add(“Data Object”))

#set ($void = $acceptedNotation.add(“Data Store”))

#set ($void = $acceptedNotation.add(“Data Input”))

#set ($void = $acceptedNotation.add(“Data Output”))

#set ($void = $acceptedNotation.add(“None Start Event”))

#set ($void = $acceptedNotation.add(“Message Start Event”))

#set ($void = $acceptedNotation.add(“Timer Start Event”))

#set ($void = $acceptedNotation.add(“Error Start Event”))

#set ($void = $acceptedNotation.add(“Escalation Start Event”))

#set ($void = $acceptedNotation.add(“Compensation Start Event”))

#set ($void = $acceptedNotation.add(“Conditional Start Event”))

#set ($void = $acceptedNotation.add(“Signal Start Event”))

#set ($void = $acceptedNotation.add(“Multiple Start Event”))

#set ($void = $acceptedNotation.add(“Parallel Multiple Start Event”))

#set ($void = $acceptedNotation.add(“None End Event”))

#set ($void = $acceptedNotation.add(“Message End Event”))

#set ($void = $acceptedNotation.add(“Error End Event”))

#set ($void = $acceptedNotation.add(“Escalation End Event”))

#set ($void = $acceptedNotation.add(“Cancel End Event”))

#set ($void = $acceptedNotation.add(“Compensation End Event”))

#set ($void = $acceptedNotation.add(“Signal End Event”))

#set ($void = $acceptedNotation.add(“Terminate End Event”))

#set ($void = $acceptedNotation.add(“Multiple End Event”))

#set ($void = $acceptedNotation.add(“Timer Boundary Event”))

#set ($void = $acceptedNotation.add(“Message Boundary Event”))

#set ($void = $acceptedNotation.add(“Error Boundary Event”))

#set ($void = $acceptedNotation.add(“Escalation Boundary Event”))

#set ($void = $acceptedNotation.add(“Cancel Boundary Event”))

#set ($void = $acceptedNotation.add(“Conditional Boundary Event”))

#set ($void = $acceptedNotation.add(“Compensation Boundary Event”))

#set ($void = $acceptedNotation.add(“Signal Boundary Event”))

#set ($void = $acceptedNotation.add(“Parallel Multiple Boundary Event”))

#set ($void = $acceptedNotation.add(“Multiple Boundary Event”))

#set ($void = $acceptedNotation.add(“Multiple Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Link Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Escalation Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Message Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Multiple Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Compensation Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Message Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Signal Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Conditional Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Parallel Multiple Catch Interemediate Event”))

#set ($void = $acceptedNotation.add(“Link Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“None Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Timer Catch Intermediate Event”))

#set ($void = $acceptedNotation.add(“Signal Throw Intermediate Event”))

#set ($void = $acceptedNotation.add(“Pool”))

#set ($void = $acceptedNotation.add(“Lane”))

#set ($void = $acceptedNotation.add(“Choreography Task”))

#set ($void = $acceptedNotation.add(“SubChoreography”))

#set ($void = $acceptedNotation.add(“Call Choreography”))

#set ($void = $acceptedNotation.add(“BPMN Collaboration”))

#set ($void = $acceptedNotation.add(“BPMN Process”))

#set ($void = $acceptedNotation.add(“Conversation”))

#set ($void = $acceptedNotation.add(“Choreography”))

#set ($void = $acceptedNotation.add(“Communication”))

#set ($void = $acceptedNotation.add(“SubConversation”))

#set ($void = $acceptedNotation.add(“Call Conversation”))

#set ($void = $acceptedNotation.add(“Initial Node”))

#set ($void = $acceptedNotation.add(“Activity Final Node”))

#set ($void = $acceptedNotation.add(“Flow Final Node”))

#set ($void = $acceptedNotation.add(“Decision Node”))

#set ($void = $acceptedNotation.add(“Fork Node”))

#set ($void = $acceptedNotation.add(“Accept Call Action”))

#set ($void = $acceptedNotation.add(“Accept Event Action”))

#set ($void = $acceptedNotation.add(“Add Structural Feature Value Action”))

#set ($void = $acceptedNotation.add(“Add Variable Value Action”))

#set ($void = $acceptedNotation.add(“Broadcast Signal Action”))

#set ($void = $acceptedNotation.add(“Call Behavior Action”))

#set ($void = $acceptedNotation.add(“Call Operation Action”))

#set ($void = $acceptedNotation.add(“Clear Association Action”))

#set ($void = $acceptedNotation.add(“Clear Structural Feature Action”))

#set ($void = $acceptedNotation.add(“Clear Variable Action”))

#set ($void = $acceptedNotation.add(“Conditional Node”))

#set ($void = $acceptedNotation.add(“Create Link Action”))

#set ($void = $acceptedNotation.add(“Create Link Object Action”))

#set ($void = $acceptedNotation.add(“Create Object Action”))

#set ($void = $acceptedNotation.add(“Destroy Link Action”))

#set ($void = $acceptedNotation.add(“Destroy Object Action”))

#set ($void = $acceptedNotation.add(“Expansion Region”))

#set ($void = $acceptedNotation.add(“Loop Node”))

#set ($void = $acceptedNotation.add(“Opaque Action”))

#set ($void = $acceptedNotation.add(“Raise Exception Action”))

#set ($void = $acceptedNotation.add(“Read Extent Action”))

#set ($void = $acceptedNotation.add(“Read Is Classified Object Action”))

#set ($void = $acceptedNotation.add(“Read Link Action”))

#set ($void = $acceptedNotation.add(“Read Link Object End Action”))

#set ($void = $acceptedNotation.add(“Read Link Object End Qualifier Action”))

#set ($void = $acceptedNotation.add(“Read Self Action”))

#set ($void = $acceptedNotation.add(“Read Structural Feature Action”))

#set ($void = $acceptedNotation.add(“Read Variable Action”))

#set ($void = $acceptedNotation.add(“Reclassify Object Action”))

#set ($void = $acceptedNotation.add(“Reduce Action”))

#set ($void = $acceptedNotation.add(“Remove Structural Feature Value Action”))

#set ($void = $acceptedNotation.add(“Remove Variable Value Action”))

#set ($void = $acceptedNotation.add(“Reply Action”))

#set ($void = $acceptedNotation.add(“Send Object Action”))

#set ($void = $acceptedNotation.add(“Send Signal Action”))

#set ($void = $acceptedNotation.add(“Sequence Node”))

#set ($void = $acceptedNotation.add(“Start Classifier Behavior Action”))

#set ($void = $acceptedNotation.add(“Structured Activity Node”))

#set ($void = $acceptedNotation.add(“Test Identity Action”))

#set ($void = $acceptedNotation.add(“Unmarshall Action”))

#set ($void = $acceptedNotation.add(“Value Specification Action”))

#set ($void = $acceptedNotation.add(“Data Store Node”))

#set ($void = $acceptedNotation.add(“Central Buffer Node”))

#set ($void = $acceptedNotation.add(“Activity Parameter Node”))

| Table of Contents |
| --- |
| Microsoft Word users please click here and press F9 to create Table of Contents.  OpenOffice.org users please remove this text and select Insert Table of Content from menu. |

| Table of Figures |
| --- |
| Microsoft Word users please click here and press F9 to create Table of Contents.  OpenOffice.org users please remove this text and select Insert Table of Content from menu. |

# Executive Summary

Most citizens don’t give much thought to the work and planning that goes into administering an election, or the process steps that election officials go through to prepare for “the big day.” It’s a complex process, and one that is under a high degree of scrutiny.

The United States has a highly decentralized election system so the process of preparing for an election varies from jurisdiction to jurisdiction, but there are commonalities. There are common process steps that every election official in the country takes.

The purpose of Election Modelling is to:

* Outline process steps so that they can be more easily understood by election officials, the media, lawmakers, and the public.
* Identify processes and associated data that can be used to develop common data formats for elections.
* Serve as a template for election administrators seeking to analyze their own processes.

# What is Process Modeling?

Processes are everywhere. There is a process to how you make your coffee, how you pump your gas, and how you order things online. Some processes are simple (e.g. pumping gas). Some are a bit more complex (e.g. buying a home). The more complex a process is the more important it is to be well documented.

A process can be performed consistenly, but only if it is well understood. Processes are often documented in natural language. Examples include laws, policies and guidelines. But natural language’s expressiveness is also its Achilles heel: it is subject to interpretation.

A key advantage of modelling notations is that they remove some of the ambiguity inherent in natural language. Notations are also useful in that they can convey information in a way that various stakeholders can understand. For example, IT professionals often need a clear, unambiguous specification to build a system as it is intended.

The election modelling work has utilized process modelling to shine a light on election processes: how they work, what they require, when they occur, and who performs them.

# Modeling Goals

The primary goal of this work is to describe processes and their associated data flows, such that use-cases for common data formats can be identified. A secondary goal of this work is to convey the current operating reality of election authorities to those outside the field. This understanding will lead to confidence in the process, as well as identification of areas for potential business process improvement.

# Methodology

The process modelling wouldn’t have been possible without the generous support of subject matter experts in the area of election administration. Election processes have been elaborated using a top down, linear time approach. These temporal categories include pre-election, election, and post-election.

## Scope

Scope answers a very simple question: “what’s included?” Scope is very important to define early on. Without this key constraint, the work loses focus and potentially its value.

There are two dimensions to scope, its breadth (what to cover) and its depth (the level of detail). The breadth has been limited to the conduct of elections in the United States by government bodies. The depth includes processes that are common across states (such as processes mandated by federal law). The model may not describe in detail state or jurisdiction specific processes, but it does its best to *accommodate* them (see [High Level, Ready for Elaboration](#_High_Level,_Ready)).

Efforts have been made to align this work with the scope of the Voluntary Voting System Guidelines (VVSG) 2.0, in development as of writing.

There is a useful distinction between core operating processes and supporting processes. Core processes are those that directly result in services provided by the election authority. There are other supporting processes that do not directly add value to the business, such as accounts payable, human resources, and information technology. Detail is provided only for operating processes. Those seeking to perform benchmarking on their operations should consider adopting the “Management and Support Processes” from APQC’s Process Classification Framework[[1]](#endnote-1).

Table 1 - Scoping Table

|  |  |  |
| --- | --- | --- |
|  | In | Out |
| Breadth | Goals of Election Systems | Everything Else |
| Depth | Common Election Processes | Divergent Election Processes |

### Descriptive vs Executable

A number of factors, including societal norms, federal and state law, and local policies inform election processes. Accommodating this patchwork of rules and regulations makes for an impossible modeling task; instead, this work has focused on the common functions an election authority is reasonably expected to perform. Such a model is considered descriptive as it provides a description of the general framework used to run the organization. It does not describe any process in enough detail for a business actor to perform (execute) his role by referring to the model alone.

### Logical vs Physical

Analogous to the descriptive vs executable distinction lies the difference between logical processes and physical processes. A logical process speaks to what is to be done, not how to do it. For example, *add voter to list of eligible voters* does not specify the form of the list. It could be a card catalog, relational database, flat file, amongst others. Where possible the process model avoids describing physical implementation details.

## High Level, Ready for Elaboration

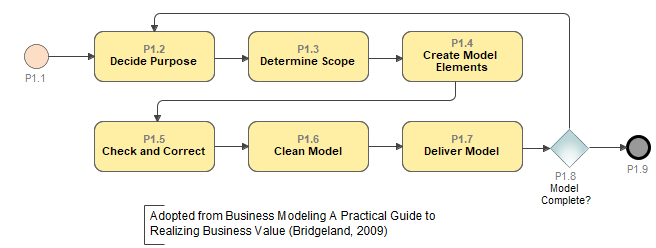
The decentralized nature of elections administration makes it very difficult to design a one size fits all model. In fact, the model is not designed to show detail at the implementation level. Instead, the model has a number of tasks that can be elaborated so that election authorities can model their specific processes and business rules.

Thus, the process model can be used as a template for election authorities wanting to model their own business processes.

## The Modeling Process

It may be easier to dive right in and start modelling, but doing so leads to many pitfalls. First of all, it is easy to become overwhelmed with complexity. The inability to see the “forest through the trees” makes models harder to understand and undermines their value. Second, a lack of formal process may lead to an uneven model, with some areas more complete than others.

Instead, a top down modelling approach is used. This methodology comes from the structured programming technique, where software is envisioned as a series of modules that can be elaborated based on functional area.



By following a top down approach to modelling, the model is always in a deliverable, albeit incomplete state.

## Workshopping

### Elicitation

Elicitation is the process of interviewing subject matter experts. The interview process involves asking questions about how elections are conducted. A wide variety of subject matter experts are necessary to ensure all perspectives are appropriately accounted for.

### Validation

Validation is the process of reviewing the models with subject matter experts to ensure the models have properly captured the essence of the election process.

## Deliverables

Key deliverables for this project include:

* A set of diagrams in Business Process Model and Notation (BPMN)[[2]](#endnote-2) notation and other notations
* A set of files describing the BPMN model in detail, including non-visual semantics.

# Creating Use-Cases from Process Model

TBD

# The Models

The election modeling work takes a holistic view of the domain of election administrations. No single model will be able to fully describe all dimensions of this enterprise. Instead, separate artifacts (diagrams) are used to answer different questions. These models roughly align with the popular Zachman Enterprise Architecture Framework [[3]](#endnote-3)perspectives:

* Executive Perspective
  + What: Glossary
* Business Management Perspective
  + What: Business Data Model
  + How: Business Process Model
  + Who: Organization Model
  + When: Business Process Model (via events)
  + Why: Business Motivation Model

## Business Process Model Diagrams

The “flowcharting” that is at the heart of the election modelling uses the Business Process Model and Notation (BPMN) notation. This standard notation will be immediately familiar to those who have done flowcharting. BPMN uses additional symbols to address some of the deficiencies of traditional flowcharting. Knowing the meaning all of these symbols is not strictly necessary to understand the election models, but it is recommended. Explaining the notation of BPMN is outside the scope of this document, however, there are numerous resources available on the internet.

### References to other models

* Swimlanes may be represented by an organization structure (e.g. role) in an Organization Structure Diagram
* Data objects may be represented by a class in a Business Data Model.

## Glossary

The result of elections in America being decentralized is that multiple lexicons have developed. This has not just resulted in different words for the same concept (synonyms), but the same words for different concepts (homonyms).

A glossary will create preferred definitions for election vocabulary and identify synonyms where they exist, with the end goal of creating a common understanding of election-related terminology.

## Business Data Model

The Business Data Model is a highly simplified data model that provides a set of data classes that can be used in business process diagrams.

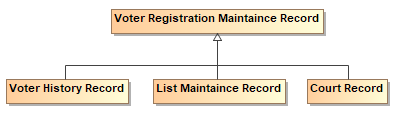


Figure 1 - A highly simplified UML Class Model

## Process Definition Diagrams

Process Definition Diagrams (sometimes called *value chains*) produce a top down catalog of the essential activities of the election authority. Each process is a puzzle piece that fits into this larger process landscape. This high-level view shows dependencies between processes and constraints placed upon them by the Business Motivation Model. Processes can be grouped into packages that highlight their interdependence.

### Example

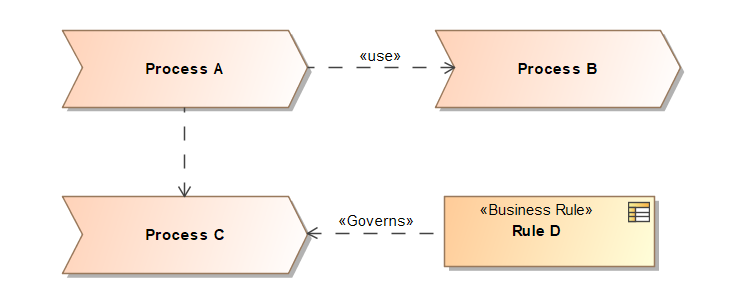


Figure 2 - Example of a Process Definition Diagram

In the above example, Process A *uses* Process B, meaning that Process B is called by Process A. Similarly, Process A *depends on* Process C, indicating that Process C likely needs to complete before Process A can start. Finally, Process C is governed by Rule D, indicating its behaviour will be constrained.

### References to other models

* Business Policies and Business Rules defined by the Business Motivation Model may constrain the process.

## Organization Structure Diagram

The Organization Structure Diagram describes the types of organizations, systems and roles that exist in the elections domain.

(The concepts described are stereotypes referencing the The Unified Profile for DoDAF/MODAF (**UPDM**)).

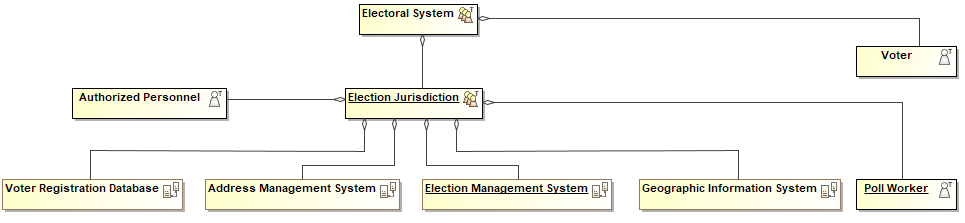


Figure 3 - Add caption here

## Business Motivation Model

The Business Motivation Model is a specification of the Object Management Group to describe the ends, means, and influencers of a business organization. In the elections domain, this allows traceability across views. For example, a legislative body may create a law (Business Policy) that is implemented as business rules that support the goals (intent) of the law.

The Business Motivation Model can then be used to constrain other views. For example, a process can only exist if it is governed by a business policy or guided by a business rule.

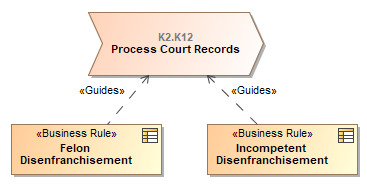


Figure 4 - A process constrained by the Business Motivation Model.

#set($packages = $array.createHashSet())

#foreach($elements in $Package)

#set($void = $packages.add($elements))

#end

#set($void = $packages.add($Package))

#foreach($elements in $report.filterElement($elements,[“Definitions”]))

#set($void = $packages.add($elements))

#end

#set($chk2=”F”)

#foreach ($package in $sorter.sort($packages, “qualifiedName”))

#foreach($diagram in $package.getOwnedDiagram())

#if($diagram.diagramType == “Business Data Diagram”)

#set($chk2=”T”)

#end

#end

#end

#if($chk2==”T”)  
##begin Value Chains

#set($valueChains = $report.filterDiagram($Diagram,["Process Definition Diagram"]))

#set($valueChainsArray = $array.createArray($valueChains))

#set($valueChain = $list.get($valueChainsArray, 0))

#if($valueChain)

# Value Chains

#foreach($diagram in $sorter.humanSort($valueChains,”owner.name”))#removeLineBreak($diagram.name)

#foreach($symbol in $report.getPresentationDiagramElements($diagram))

#set($elm = $symbol.getElement())

#foreach($stereotype in $report.getStereotypes($elm))

#if(!$report.isEmpty($stereotype.icon))

#set($void = $stereotypesInDiagram.add($stereotype))

#end

#end

#end

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#end

#end

##END VALUE CHAINS

#set($BPMNprocesses = $report.filterElement($elements, [“BPMN Process”]))

#set($BPMNprocessesArray = $array.createArray($BPMNprocesses))

#set($BPMNprocess = $list.get($BPMNprocessesArray, 0))

#if($BPMNprocess)

# Processes

**Resources in Processes:**

| **Process Name** | **Id** | **Related Resources** |
| --- | --- | --- |
| #forrow($process in $sorter.humanSort($BPMNprocesses,”id”)) $process.name | $process.id | #foreach($element in $process.getOwnedElement())  #if($report.getAppliedStereotypeByName($element,”LaneSet”))  #if($report.getAppliedStereotypeByName($element.represents, “Resource”))   * $report.getIconFor($element.represents)$element.represents.name   #end  #foreach($lane in $element.getSubpartition())  #if($report.getAppliedStereotypeByName($lane.represents, “Resource”))   * $report.getIconFor($lane.represents)$lane.represents.name   #end  #if($lane.getSubpartition().size()>0)  #foreach($inner in $lane.getSubpartition())  #if($report.getAppliedStereotypeByName($inner.represents, “Resource”))   * $report.getIconFor($inner.represents)$inner.represents.name   #end  #end  #end  #end  #end  #end  #endrow |

#end

#set($BPMNprocesses = $report.filterElement($elements, [“BPMN Process”]))

#set($BPMNprocessesArray = $array.createArray($BPMNprocesses))

#set($BPMNprocess = $list.get($BPMNprocessesArray, 0))

#if($BPMNprocess)

#foreach($BPMNprocess in $sorter.sort($BPMNprocesses, “id:asc”))

#set($mainDiagram = $array.createArray())

#set($innerDiagram = $array.createArray())

#set($void = $array.addCollection($mainDiagram,$BPMNprocess.ownedDiagram))

#diagrams($BPMNprocess)

#macro (diagrams $element)

#foreach($elm in $element.getOwnedElement())

##LEGEND SUPPORT JND (3/18)

#foreach($stereotype in $report.getStereotypes($elm))

##Check for stereotype icon existing

#if(!$report.isEmpty($stereotype.icon))

#set($void = $stereotypesInDiagram.add($stereotype))

#end

#end

#if($elm.elementType==”subprocess”)

#if($elm.ownedDiagram)

#set($void = $array.addCollection($innerDiagram,$elm.ownedDiagram))

#diagrams($elm)

#end

#end

#end

#end

## **Process $BPMNprocess.id $bookmark.create($BPMNprocess.ID,$BPMNprocess.name)**

#if($BPMNprocess.documentation)

$BPMNprocess.documentation

#end

#set($void = $sorter.sort($mainDiagram,’name’))

#set($void = $sorter.sort($innerDiagram,’name’))

#set($void = $array.addCollection($mainDiagram,$innerDiagram))

#foreach($diagram in $mainDiagram)

#if ($diagram.diagramType == ”BPMN Process Diagram”)

#removeLineBreak($diagram.name)

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#if ($diagram.documentation != ‘’)

$diagram.documentation

#end

$group.init()

#set ($elementList = ‘’)

#set ($elementList = $array.createArray())

#foreach ($obj in $report.getDiagramElements($diagram))

#resolveNotationName($obj)

#if ($acceptedNotation.contains($resolveNotationName))$group.put($resolveNotationName, $obj)

#set($void = $elementList.add($obj))#end

#end

#set ($sortedElementList = ‘’)

#set ($sortedElementList = $array.createArray())

#foreach ($notationName in $sorter.sort($group.groupNames()))

#set ($acceptedelements = $group.get($notationName))

#foreach ($element in $sorter.sort($acceptedelements))

#set ($void = $sortedElementList.add($element))

#end

#end

#set($activityList=$array.createArray())

#set($itemAwareList=$array.createArray())

#foreach($element in $sortedElementList)

#if($element.elementType==”task”||$element.elementType==”sendtask”||$element.elementType==”receivetask”||$element.elementType==”servicetask”||$element.elementType==”scripttask”||$element.elementType==”manualtask”||$element.elementType==”businessruletask”||$element.elementType==”usertask”||$element.elementType==”subprocess”||$element.elementType==”transactionsubprocess”||$element.elementType==”adhocsubprocess”||$element.elementType==”callactivity”)

#set($void=$activityList.add($element))

#elseif($element.elementType==”dataobject”|| $element.elementType==”datastore”|| $element.elementType==”datainput”|| $element.elementType==”dataoutput”)

#set($void=$itemAwareList.add($element))

#end

#end

#if($activityList.size()>0)

**Process Activities:**

| **Name** | **Description** | **Properties** |
| --- | --- | --- |
| #forrow($element in $sorter.humanSort($activityList,"id"))$report.getIconFor($element) $element.id $element.name | $element.documentation | #if(!$report.isEmpty($element.assignedResources ))  Resources:  #foreach($rr in $element.assignedResources)   * $report.getIconFor($rr) $bookmark.open($rr.ID,$rr.name)   #end  #end  #if($element.elementType==”callactivity”&&$element.behavior)  Called Element:   * $report.getIconFor($element.behavior) $bookmark.open( $element.behavior.ID,$element.behavior.name)   #end  #endrow |

#end

#if($itemAwareList.size()>0)

**Data used in Process:**

| **Name** | **Description** |
| --- | --- |
| #forrow($element in $sorter.humanSort($itemAwareList,"id"))#if($element.type) $report.getIconFor($element) $element.name [$report.getIconFor($element.type) $element.type.name]  #else  $report.getIconFor($element) $element.name  #end | #if($element.documentation!=’’)  $report.getIconFor($element)$element.name - $element.documentation  #end  #if($element.type&&$element.type.documentation!='')  $report.getIconFor($element.type) $element.type.name - $element.type.documentation  #end  #endrow |

#end

#end

#end

#end

#end

#set($BPMNcollaborations = $report.filterElement($elements, [“BPMN Collaboration”]))

#set($BPMNcollaborationsArray = $array.createArray($BPMNcollaborations))

#set($BPMNcollaboration = $list.get($BPMNcollaborationsArray, 0))

#if($BPMNcollaboration)

# Collaborations

**Resources in Collaborations:**

| **Name** | **Related Resources** |
| --- | --- |
| #forrow($colla in $sorter.humanSort($BPMNcollaborations,”id”))$colla.name | #foreach($element in $colla.getOwnedElement())  #if($report.getAppliedStereotypeByName($element, “LaneSet”))  #if($report.getAppliedStereotypeByName($element.represents, “Resource”))   * $report.getIconFor($element.represents) $element.represents.name   #foreach($lane in $element.getSubpartition())  #if($report.getAppliedStereotypeByName($lane.represents, “Resource”))   * $report.getIconFor($lane.represents) $lane.represents.name   #if($lane.getSubpartition().size()>0)  #foreach($inner in $lane.getSubpartition())  #if($report.getAppliedStereotypeByName($inner.represents, “Resource”))   * $report.getIconFor($inner.represents) $inner.represents.name   #end  #end  #end  #end  #end  #end  #end  #end  #endrow |

#end

#set($BPMNcollaborations = $report.filterElement($elements, [“BPMN Collaboration”]))

#set($BPMNcollaborationsArray = $array.createArray($BPMNcollaborations))

#set($BPMNcollaboration = $list.get($BPMNcollaborationsArray, 0))

#if($BPMNcollaboration)

#foreach($BPMNcollaboration in $sorter.sort($BPMNcollaborations,”id:asc”))

#set($mainDiagram = $array.createArray())

#set($innerDiagram = $array.createArray())

#set($void = $array.addCollection($mainDiagram,$BPMNcollaboration.ownedDiagram))

#foreach($elm in $BPMNcollaboration.getOwnedElement())

#if($elm.elementType==”subprocess”)

#if($elm.ownedDiagram)

#set($void = $array.addCollection($innerDiagram,$elm.ownedDiagram))

#end

#end

#end

## **Collaboration $BPMNcollaboration.name**

#if($BPMNcollaboration.documentation)

$BPMNcollaboration.documentation

#end

#set($void = $sorter.sort($mainDiagram,’name’))

#set($void = $sorter.sort($innerDiagram,’name’))

#set($void = $array.addCollection($mainDiagram,$innerDiagram))

#foreach($diagram in $mainDiagram)

#if ($diagram.diagramType == “BPMN Collaboration Diagram” || $diagram.diagramType == “BPMN Process Diagram”)

#foreach($symbol in $report.getPresentationDiagramElements($diagram))

#set($element = $symbol.getElement())

#foreach($stereotype in $report.getStereotypes($element))

##Check for stereotype icon existing

#if(!$report.isEmpty($stereotype.icon))

#set($void = $stereotypesInDiagram.add($stereotype))

#end

#end

#end

#removeLineBreak($diagram.name)

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#if ($diagram.documentation != ‘’)

$diagram.documentation

#end

$group.init()

#set ($elementList = ‘’)

#set ($elementList = $array.createArray())

#foreach ($obj in $report.getDiagramElements($diagram))

#resolveNotationName($obj)

#if ($acceptedNotation.contains($resolveNotationName))$group.put($resolveNotationName, $obj)

#set($void = $elementList.add($obj))#end

#end

#set($poolList=$array.createArray())

#set($convList=$array.createArray())

#foreach ($notationName in $sorter.sort($group.groupNames()))

#set ($acceptedelements = $group.get($notationName))

#foreach ($element in $sorter.sort($acceptedelements))

#if($element.elementType==”conversation”||$element.elementType==”subconversation”||$element.elementType==”callconversation”)

#set ($void = $convList.add($element))

#elseif($element.elementType==”pool”)

#set ($void = $poolList.add($element))

#end

#end

#end

#if($poolList.size()>0)

**Message Flow description:**

| **Name** | **MessageFlow** |
| --- | --- |
| #forrow($element in $sorter.sort($poolList, 'name'))  $report.getIconFor($element) $element.name | #foreach($mf in $sorter.sort($element.outgoingMessageFlows,’name’))  #foreach($tg in $mf.supplier)  #if($tg.elementType==”pool”)  #if($tg.represents)  #set($target=$tg.represents)  #else  #set($target=$tg)  #end  #else  #set($target=$tg)  #end  #end  #if($mf.messageRef)  #set($msg=$mf.messageRef)  Sent Message **$msg.name** to $report.getIconFor($target) $target.name  #else  Message Flow **$mf.name** to $report.getIconFor($target) $target.name  #end  #end  #endrow |

#end

#if($convList.size()>0)

**Conversation Node description:**

| **Name** | **Description** | **MessageFlowRefs** |
| --- | --- | --- |
| #forrow($element in $sorter.sort($convList, 'name'))$report.getIconFor($element) $element.name | $element.documentation | #foreach($mf in $element.messageFlowRefs)  #foreach($src in $mf.client)  #set($source=$src)  #end  #foreach($tg in $mf.supplier)  #set($target=$tg)  #end   * Message Flow **$mf.name** [$report.getIconFor($source) $source.name - $report.getIconFor($target) $target.name]   #end  #endrow |

#end

#end

#end

#end

#end

#set($BPMNchoreographies = $report.filterElement($elements, [“Choreography”]))

#set($BPMNchoreographiesArray = $array.createArray($BPMNchoreographies))

#set($BPMNchoreography = $list.get($BPMNchoreographiesArray, 0))

#if($BPMNchoreography)

# Choreographies

**Participants in Choreographies:**

| **Name** | **Related Participants** |
| --- | --- |
| #forrow($choreo in $sorter.humanSort($BPMNchoreographies,”name”)) $report.getIconFor($choreo) $choreo.name | #set($participantList=$array.createHashSet())  #foreach($element in $choreo.getOwnedElement())  #if($element.elementType==”choreographytask”||$element.elementType=="subchoreography"||$element.elementType=="callchoreography")  #foreach($par in $element.participants)  #set($void=$participantList.add($par))  #end  #end  #end  #if($participantList.size()>0)  #foreach($par in $sorter.sort($participantList,”name”))   * $report.getIconFor($par) $par.name   #end  #end  #endrow |

#end

#set($BPMNchoreographies = $report.filterElement($elements, [“Choreography”]))

#set($BPMNchoreographiesArray = $array.createArray($BPMNchoreographies))

#set($BPMNchoreography = $list.get($BPMNchoreographiesArray, 0))

#if($BPMNchoreography)

#foreach($BPMNchoreography in $sorter.sort($BPMNchoreographies,”id:asc”))

#set($mainDiagram = $array.createArray())

#set($innerDiagram = $array.createArray())

#set($void = $array.addCollection($mainDiagram,$BPMNchoreography.ownedDiagram))

#foreach($elm in $BPMNchoreography.getOwnedElement())

#if($elm.elementType==”subchoreography”)

#if($elm.ownedDiagram)

#set($void = $array.addCollection($innerDiagram,$elm.ownedDiagram))

#end

#end

#end

## **Choreography $BPMNchoreography.name**

#if($BPMNchoreography.documentation != ‘’)

$BPMNchoreography.documentation

#end

#set($void = $sorter.sort($mainDiagram,’name’))

#set($void = $sorter.sort($innerDiagram,’name’))

#set($void = $array.addCollection($mainDiagram,$innerDiagram))

#foreach($diagram in $mainDiagram)

#if ($diagram.diagramType == “BPMN Choreography Diagram”)

#removeLineBreak($diagram.name)

#foreach($symbol in $report.getPresentationDiagramElements($diagram))

#set($element = $symbol.getElement())

#foreach($stereotype in $report.getStereotypes($element))

##Check for stereotype icon existing

#if(!$report.isEmpty($stereotype.icon))

#set($void = $stereotypesInDiagram.add($stereotype))

#end

#end

#end

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#if ($diagram.documentation != ‘’)

$diagram.documentation

#end

$group.init()

#set ($elementList = ‘’)

#set ($elementList = $array.createArray())

#foreach ($obj in $report.getDiagramElements($diagram))

#resolveNotationName($obj)

#if ($acceptedNotation.contains($resolveNotationName))$group.put($resolveNotationName, $obj)

#set($void = $elementList.add($obj))#end

#end

#set ($sortedElementList = ‘’)

#set ($sortedElementList = $array.createArray())

#foreach ($notationName in $sorter.sort($group.groupNames()))

#set ($acceptedelements = $group.get($notationName))

#foreach ($element in $sorter.sort($acceptedelements))

#if($element.elementType==”choreographytask”|| $element.elementType==”subchoreography”|| $element.elementType==”callchoreography”)

#set ($void = $sortedElementList.add($element))

#end

#end

#end

#if($sortedElementList.size()>0)

**Choreography Activities:**

| **Name** | **Description** | **Related elements** |
| --- | --- | --- |
| #forrow($element in $sorter.humanSort($sortedElementList, “name”))  $report.getIconFor($element) $element.name | $element.documentation | #if($element.elementType==”choreographytask”|| $element.elementType==”subchoreography”|| $element.elementType==”callchoreography”)  #if($element.participants)  #set($init=$element.initiatingParticipant)  #end  #if($element.participants)  #foreach($par in $element.participants)  #if($init.name == $par.name)  $report.getIconFor($par) $par.name [Initiating]  #else  $report.getIconFor($par) $par.name  #end#end#end#end#endrow |

#end

#end

#end

#end

#end

# Business Data

#foreach ($package in $sorter.sort($packages, “qualifiedName”))

#set($chk=”F”)

#foreach($diagram in $package.getOwnedDiagram())

#if($diagram.diagramType == “Business Data Diagram”)

#set($chk=”T”)

#end

#end

#if($chk==”T”)

#foreach($diagram in $sorter.sort($package.getOwnedDiagram(),’name’))

#if($diagram.diagramType == “Business Data Diagram”)

#removeLineBreak($diagram.name)

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#if ($diagram.documentation != ‘’)

$diagram.documentation

#end

#set($checkBusinessElement=”F”)

#set ($classList = $array.createArray())

#set ($enumerationList = $array.createArray())

#foreach($element in $report.getDiagramElements($diagram))

#if($element.elementType == ’class’)

#set($void = $classList.add($element))

#set($checkBusinessElement=”T”)

#elseif($element.elementType == ’enumeration’)

#set($void = $enumerationList.add($element))

#set($checkBusinessElement=”T”)

#end

#end

#if($checkBusinessElement==”T”)

#if($classList.size() > 0)

**Business Data Class descriptions:**

| **Name** | **Description** | **Attributes** |
| --- | --- | --- |
| #forrow($element in $sorter.sort($classList,”name”))  $element.name | $element.documentation | #foreach ($att in $element.getOwnedAttribute())   * $att.name #if($att.documentation!=’’)- $att.documentation#end   #end#endrow |

#end

#if($enumerationList.size() > 0)

**Enumeration descriptions:**

| **Name** | **Description** | **Enumeration Literals** |
| --- | --- | --- |
| #forrow($element in $sorter.sort($enumerationList,”name”))  $element.name | $element.documentation | #foreach ($lit in $element.getOwnedLiteral())   * $lit.name #if($lit.documentation!=’’)- $lit.documentation#end   #end#endrow |

#end

#end

#end

#end

#end

#end

#end

#set($packages = $array.createHashSet())

#foreach($elements in $Package)

#set($void = $packages.add($elements))

#end

#set($void = $packages.add($Package))

#foreach($elements in $report.filterElement($elements,[“Definitions”]))

#set($void = $packages.add($elements))

#end

#set($chk2=”F”)

#foreach ($package in $sorter.sort($packages, “qualifiedName”))

#foreach($diagram in $package.getOwnedDiagram())

#if($diagram.diagramType == “Organization Structure Diagram”)

#set($chk2=”T”)

#end

#end

#end

#set($checkPar=”F”)

#set($allpar = $report.filterElement($elements,[“Participant”]))

#set($participants = $array.createArray())

#foreach($participant in $allpar)

#if(!$report.getAppliedStereotypeByName($participant, “Organization”))

#set($void = $participants.add($participant))

#set($checkPar=”T”)

#end

#end

#set($allResourceRole= $report.filterElement($elements,[“Resource”]))

#set($resourceRoles = $array.createArray())

#foreach($rr in $allResourceRole)

#if(!($report.getAppliedStereotypeByName($rr, “Post”)||$report.getAppliedStereotypeByName($rr, “Person”)))

#set($void = $resourceRoles.add($rr))

#set($checkPar=”T”)

#end

#end

#foreach($elements in $report.filterElement($elements,[“Performer”,”Human Performer”,”Potential Owner”]))

#set($checkPar=”T”)

#end

#if($chk2==”T”||$checkPar==”T”)

# Organization Structure

**#if($checkPar==”T”)**

#set($Resources = $report.filterElement($elements, [“Resource”]))

#set($ResourcesArray = $array.createArray($Resources))

#if(!$report.isEmpty($ResourcesArray))

**Resource descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($resource in $sorter.humanSort($ResourcesArray ,’name’))  $report.getIconFor($resource) $bookmark.create($resource.ID,$resource.name) | $resource.documentation #endrow |

#end

#set($Organizations = $report.filterElement($elements, [“Organization Unit”]))

#set($OrganizationArray = $array.createArray($Organizations))

#if(!$report.isEmpty($OrganizationArray))

**Organization descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($org in $sorter.humanSort($OrganizationArray,’name’))  $report.getIconFor($org) $org.name | $org.documentation #endrow |

#end

#set($Roles = $report.filterElement($elements, [“Role”]))

#set($RoleArray = $array.createArray($Roles))

#if(!$report.isEmpty($RoleArray))

**Role descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($role in $sorter.humanSort($RoleArray,’name’))  $report.getIconFor($role) $role.name | $role.documentation #endrow |

#end

#set($Persons = $report.filterElement($elements, [“Person”]))

#set($PersonArray = $array.createArray($Persons))

#if(!$report.isEmpty($PersonArray))

**Person descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($person in $sorter.humanSort($PersonArray,’name’))  $report.getIconFor($person) $person.name | $person.documentation #endrow |

#end

#set($IS = $report.filterElement($elements, [“Information System”]))

#set($ISArray = $array.createArray($IS))

#if(!$report.isEmpty($ISArray))

**Information system descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($is in $sorter.humanSort($ISArray,’name’))  $report.getIconFor($is) $is.name | $is.documentation #endrow |

#end

#end

#foreach ($package in $sorter.sort($packages, “qualifiedName”))

#set($chk=”F”)

#foreach($diagram in $package.getOwnedDiagram())

#if($diagram.diagramType == “Organization Structure Diagram”)

#set($chk=”T”)

#end

#end

#if($chk==”T”)

#foreach($diagram in $package.getOwnedDiagram())

#if($diagram.diagramType == “Organization Structure Diagram”)

#removeLineBreak($diagram.name)

#foreach($symbol in $report.getPresentationDiagramElements($diagram))

#set($element = $symbol.getElement())

#foreach($stereotype in $report.getStereotypes($element))

##Check for stereotype icon existing

#if(!$report.isEmpty($stereotype.icon))

#set($void = $stereotypesInDiagram.add($stereotype))

#end

#end

#end

### **Diagram $removeLineBreak**

$diagram.image

1. $removeLineBreak

#if ($diagram.documentation != ‘’)

$diagram.documentation

#end

#set($node = $array.createArray())

#foreach($elm in $report.getDiagramElements($diagram))

#if($report.getAppliedStereotypeByName($elm,”Organization”)||$report.getAppliedStereotypeByName($elm,”Post”)||$report.getAppliedStereotypeByName($elm,”Person”) ||$report.getAppliedStereotypeByName($elm,”InformationSystem”))

#set($void = $node.add($elm))

#end

#end

#if(!$report.isEmpty($node))

**Organization Element descriptions:**

| **Name** | **Description** |
| --- | --- |
| #forrow($element in $sorter.humanSort($node,’appliedStereotype’))  $report.getIconFor($element) $element.name | $element.documentation#endrow |

#end

#end

#end

#end

#end

#end

# Legend

**Used Symbols:**

| **Symbol** | **Name** | **Description** |
| --- | --- | --- |
| #forrow($stereotype in $sorter.humanSort($stereotypesInDiagram, “name"))  ##Check for stereotype icon existing  #if(!$report.isEmpty($stereotype.icon))  ##Print icon and name of stereotype  $report.getIconFor($stereotype)  #end | $stereotype.name | $stereotype.documentation #endrow |

1. AQPC Process Goes Here [↑](#endnote-ref-1)
2. http://www.omg.org/spec/BPMN [↑](#endnote-ref-2)
3. https://www.zachman.com/about-the-zachman-framework [↑](#endnote-ref-3)