Document: Primitive Action Metadata

I. Objective

This document describes the model of primitive action metadata (PAM) for generating data elasticity management process, as well as GUI to extend PAM.

II. Model of Primitive Action Metadata

a. Concept

A primitive action can be a monitoring action, adjustment action or resource control action. A monitoring action is used to understand QoR of a data asset. An adjustment action is used to adjust QoR of a data asset. A resource control action is used to manage the resource usage at infrastructure.

b. UML

i. Monitoring Action

Figure 1 shows the model of monitoring action. A monitoring action is used to monitor quality of data. Calling monitoring actions will returns the values of quality of data assets.
The `MonitoringAction` class has:

- `Artifact`: software artifact includes its location (i.e., a direct link to download the artifact), type (i.e., can be a shell script, a jar file or a war file), restfulAPI (i.e., REST API to call its corresponding service after deployment) and httpMethod (i.e., the method of REST API can be PUT, GET and POST).
- `AssociatedQoRMetric`: the QoR metric of a data asset can be monitored by using this monitoring action.
- `listOfParameters`: list of parameters for this monitoring action.

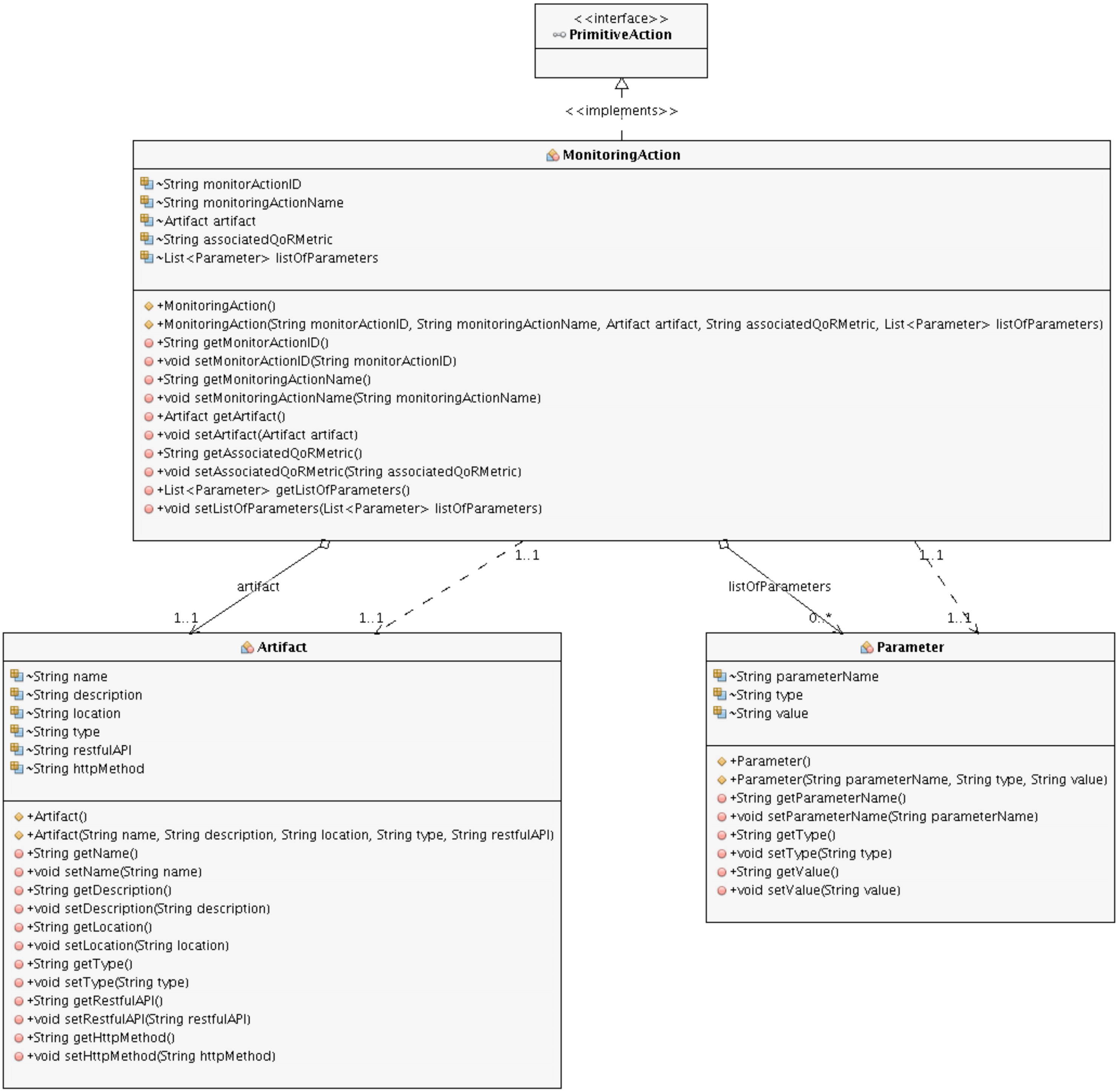

<<interface>>
PrimitiveAction
<<implements>>
MonitoringAction
~String monitorActionID
~String monitoringActionName
~Artifact artifact
~String associatedQoRMetric
~List<Parameter> listOfParameters
+MonitoringAction()
+MonitoringAction(String monitorActionID, String monitoringActionName, Artifact artifact, String associatedQoRMetric, List<Parameter> listOfParameters)
+String getMonitorActionID()
+void setMonitorActionID(String monitorActionID)
+String getMonitoringActionName()
+void setMonitoringActionName(String monitoringActionName)
+Artifact getArtifact()
+void setArtifact(Artifact artifact)
+String getAssociatedQoRMetric()
+void setAssociatedQoRMetric(String associatedQoRMetric)
+List<Parameter> getListOfParameters()
+void setListOfParameters(List<Parameter> listOfParameters)
artifact
1..1
1..1
1..1
listOfParameters
0..*
1..1
1..1
Artifact
~String name
~String description
~String location
~String type
~String restfulAPI
~String httpMethod
+Artifact()
+Artifact(String name, String description, String location, String type, String restfulAPI)
+String getName()
+void setName(String name)
+String getDescription()
+void setDescription(String description)
+String getLocation()
+void setLocation(String location)
+String getType()
+void setType(String type)
+String getRestfulAPI()
+void setRestfulAPI(String restfulAPI)
+String getHttpMethod()
+void setHttpMethod(String httpMethod)
Parameter
~String parameterName
~String type
~String value
+Parameter()
+Parameter(String parameterName, String type, String value)
+String getParameterName()
+void setParameterName(String parameterName)
+String getType()
+void setType(String type)
+String getValue()
+void setValue(String value)

Figure 1: Model of Monitoring Action

ii. Adjustment Action

Figure 2 shows the model of an adjustment action. The class `AdjustmentAction` has:

- `Artifact`: the description in the previous subsection
- `AssociatedQoRMetric`: the QoR metric can be adjusted by this adjustment action
- `listOfPrerequisiteActionIDs`: the list of adjustment actions need to be executed before this action to avoid data corruption and data overwritten
- `listOfAdjustmentCases`: To deal with different cases of quality of a data asset, an adjustment action can have multiple adjustment cases. An adjustment case has an estimatedResult(i.e., a range of values of this QoR metric can be achieved by using this action with the parameters of this case), analyticTask (i.e., analytic task and and its parameters in PAM is used to match with ones in data analytic functions to indicate a right adjustment case) and listOfParameters (i.e., parameters are used for this adjustment case)

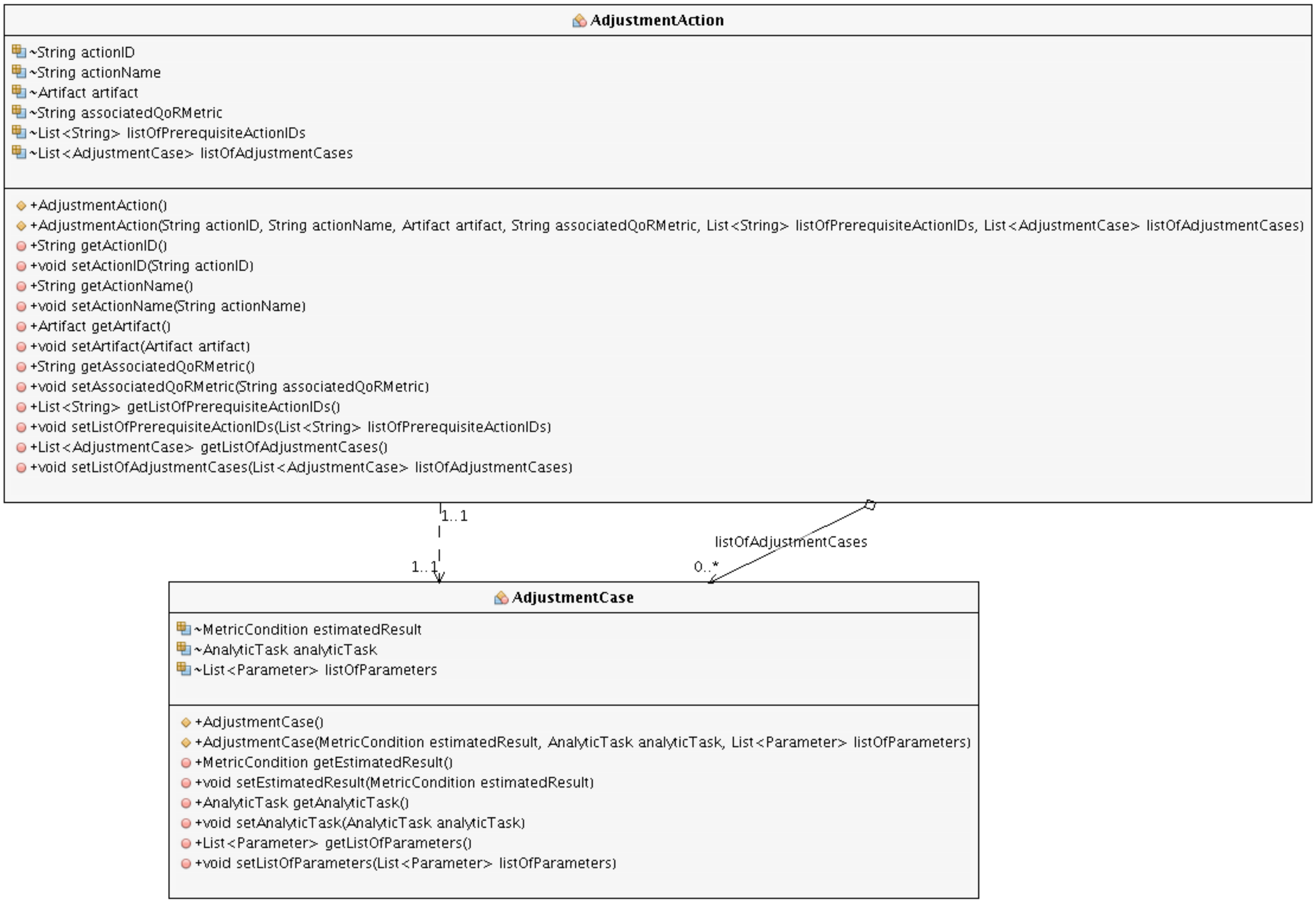

AdjustmentAction
~String actionID
~String actionName
~Artifact artifact
~String associatedQoRMetric
~List<String> listOfPrerequisiteActionIDs
~List<AdjustmentCase> listOfAdjustmentCases
+AdjustmentAction()
+AdjustmentAction(String actionID, String actionName, Artifact artifact, String associatedQoRMetric, List<String> listOfPrerequisiteActionIDs, List<AdjustmentCase> listOfAdjustmentCases)
+String getActionID()
+void setActionID(String actionID)
+String getActionName()
+void setActionName(String actionName)
+Artifact getArtifact()
+void setArtifact(Artifact artifact)
+String getAssociatedQoRMetric()
+void setAssociatedQoRMetric(String associatedQoRMetric)
+List<String> getListOfPrerequisiteActionIDs()
+void setListOfPrerequisiteActionIDs(List<String> listOfPrerequisiteActionIDs)
+List<AdjustmentCase> getListOfAdjustmentCases()
+void setListOfAdjustmentCases(List<AdjustmentCase> listOfAdjustmentCases)
1..1
1..1
listOfAdjustmentCases
0..*
AdjustmentCase
~MetricCondition estimatedResult
~AnalyticTask analyticTask
~List<Parameter> listOfParameters
+AdjustmentCase()
+AdjustmentCase(MetricCondition estimatedResult, AnalyticTask analyticTask, List<Parameter> listOfParameters)
+MetricCondition getEstimatedResult()
+void setEstimatedResult(MetricCondition estimatedResult)
+AnalyticTask getAnalyticTask()
+void setAnalyticTask(AnalyticTask analyticTask)
+List<Parameter> getListOfParameters()
+void setListOfParameters(List<Parameter> listOfParameters)

Figure 2: Model of Adjustment Action

iii. Resource Control Action

Figure 3 shows the model of resource control action. The class `ResourceControlAction` has:

- `AssociatedQoRMetric`: the QoR metric can be managed by this resource control action
- `listOfResourceControlCases`: to deal with different cases of resource usage control to achieve specific values of data asset, a resource control action can have multiple resource control cases. A `ResourceControlCase` indicated by `estimatedResult` and `analyticTask`, similar to ones in adjustment case as described in previous subsection. A `ResourceControlCase` has a `listOfResourceControlStrategies`. A resource control strategy has `controlMetric` (i.e., metric can be controlled at infrastructure such cpuUsage), `primitiveAction` (i.e., the corresponding service of this action will be applied this control strategy) and the `scaleInCondition` & `scaleOutCondition` are used to make decision to scale the corresponding service when these conditions are met.

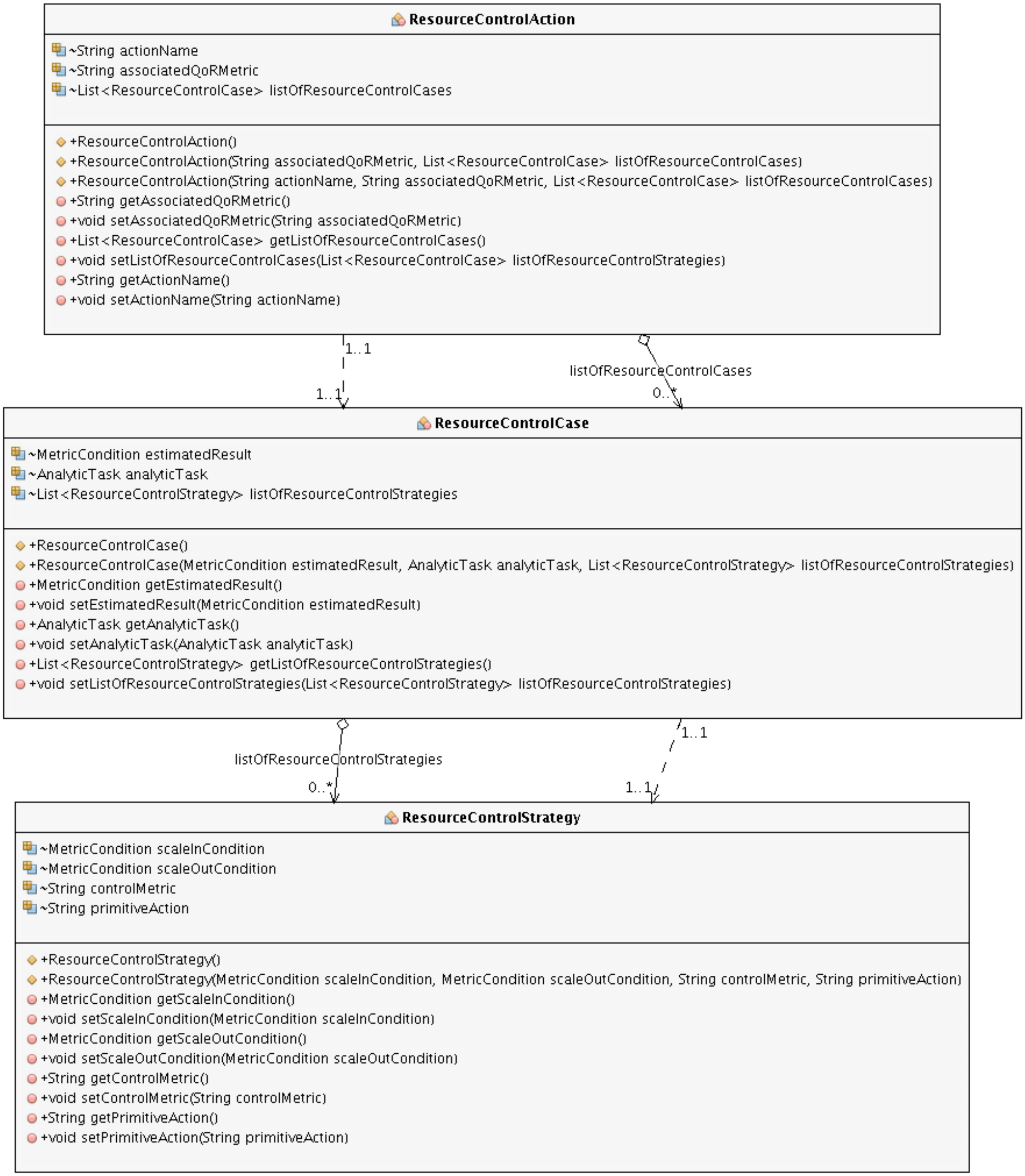

ResourceControlAction
~String actionName
~String associatedQoRMetric
~List<ResourceControlCase> listOfResourceControlCases
+ResourceControlAction()
+ResourceControlAction(String associatedQoRMetric, List<ResourceControlCase> listOfResourceControlCases)
+ResourceControlAction(String actionName, String associatedQoRMetric, List<ResourceControlCase> listOfResourceControlCases)
+String getAssociatedQoRMetric()
+void setAssociatedQoRMetric(String associatedQoRMetric)
+List<ResourceControlCase> getListOfResourceControlCases()
+void setListOfResourceControlCases(List<ResourceControlCase> listOfResourceControlStrategies)
+String getActionName()
+void setActionName(String actionName)
1..1
1..1
listOfResourceControlCases
0..*
ResourceControlCase
~MetricCondition estimatedResult
~AnalyticTask analyticTask
~List<ResourceControlStrategy> listOfResourceControlStrategies
+ResourceControlCase()
+ResourceControlCase(MetricCondition estimatedResult, AnalyticTask analyticTask, List<ResourceControlStrategy> listOfResourceControlStrategies)
+MetricCondition getEstimatedResult()
+void setEstimatedResult(MetricCondition estimatedResult)
+AnalyticTask getAnalyticTask()
+void setAnalyticTask(AnalyticTask analyticTask)
+List<ResourceControlStrategy> getListOfResourceControlStrategies()
+void setListOfResourceControlStrategies(List<ResourceControlStrategy> listOfResourceControlStrategies)
listOfResourceControlStrategies
0..*
1..1
1..1
ResourceControlStrategy
~MetricCondition scaleInCondition
~MetricCondition scaleOutCondition
~String controlMetric
~String primitiveAction
+ResourceControlStrategy()
+ResourceControlStrategy(MetricCondition scaleInCondition, MetricCondition scaleOutCondition, String controlMetric, String primitiveAction)
+MetricCondition getScaleInCondition()
+void setScaleInCondition(MetricCondition scaleInCondition)
+MetricCondition getScaleOutCondition()
+void setScaleOutCondition(MetricCondition scaleOutCondition)
+String getControlMetric()
+void setControlMetric(String controlMetric)
+String getPrimitiveAction()
+void setPrimitiveAction(String primitiveAction)

Figure 3: The Model of Resource Control Action

III. Graphical User Interface

a. PAM Repository

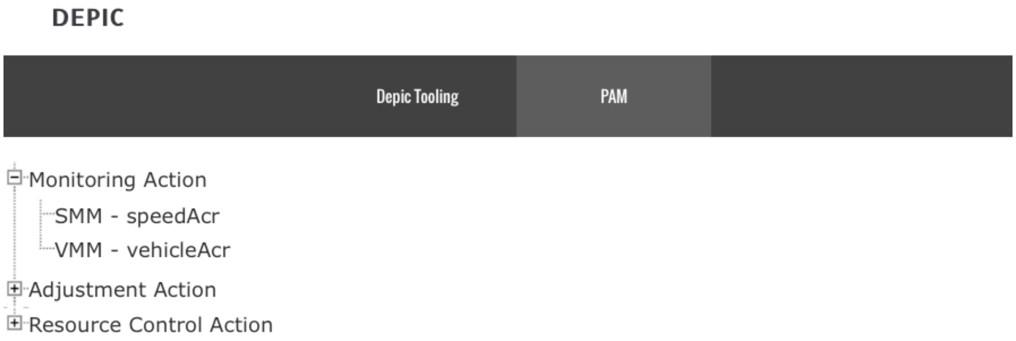
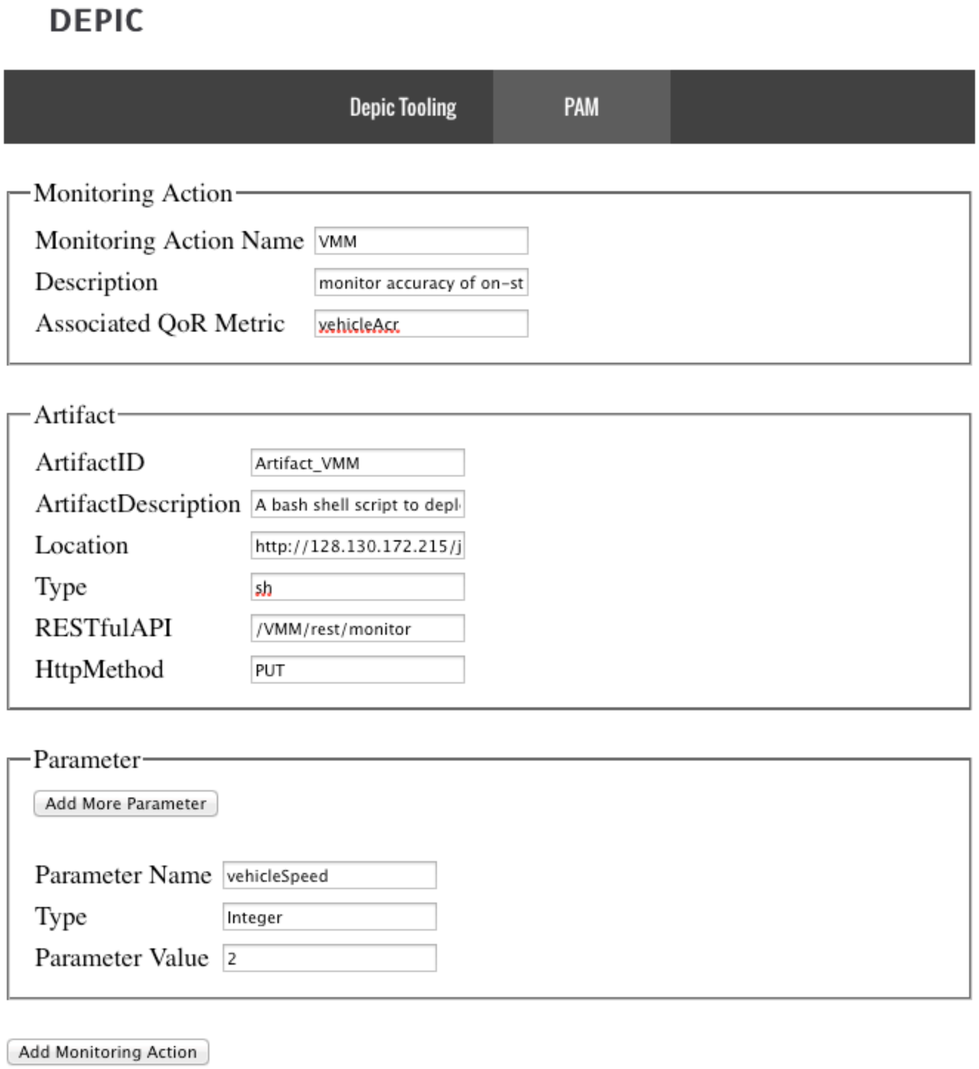


Figure 4: View primitive actions

b. Adding Monitoring Action



Monitoring Action

Monitoring Action Name	VMM
Description	monitor accuracy of on-st
Associated QoR Metric	vehicleAcr

Artifact

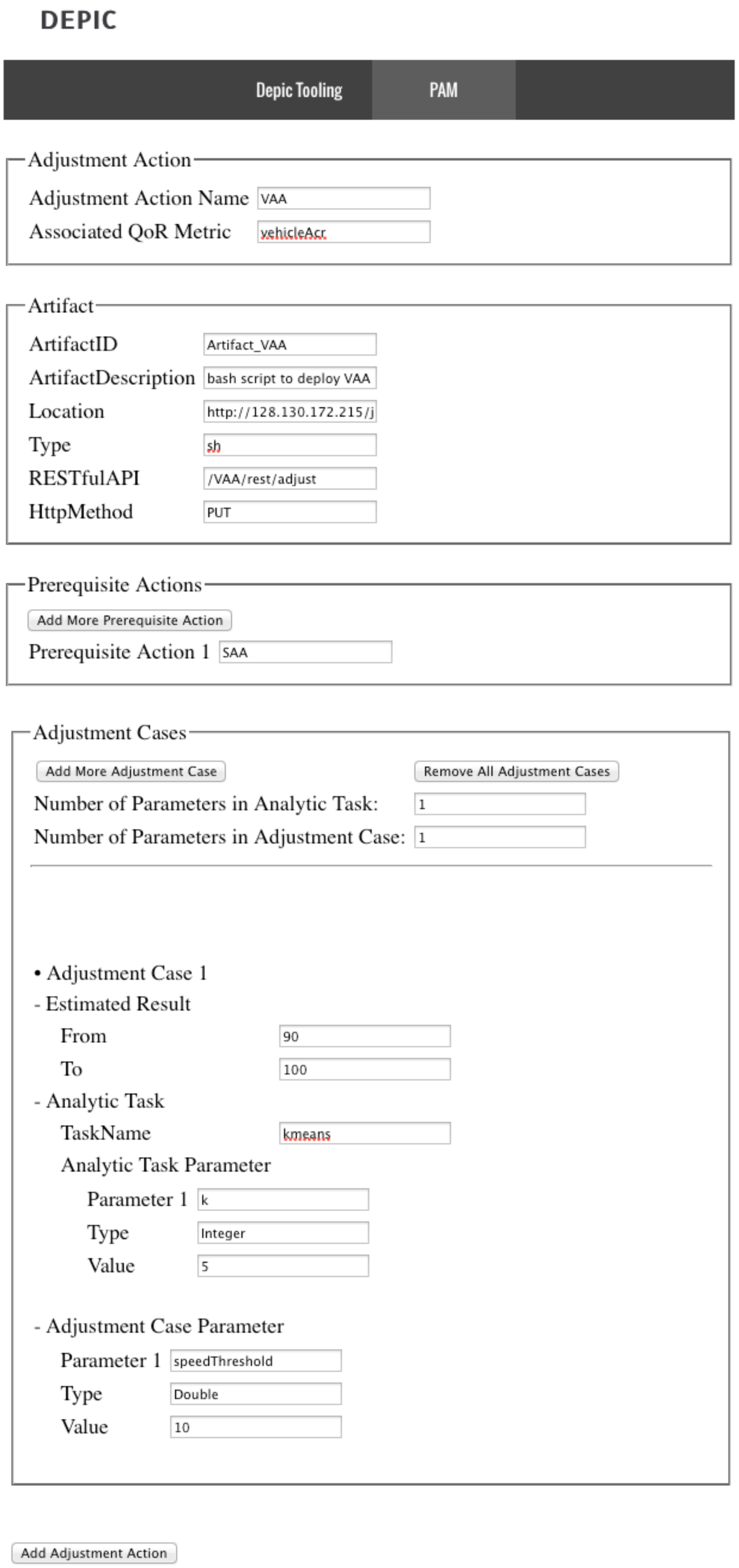
ArtifactID	Artifact_VMM
ArtifactDescription	A bash shell script to depl
Location	http://128.130.172.215/j
Type	sh
RESTfulAPI	/VMM/rest/monitor
HttpMethod	PUT

Parameter

Parameter Name	vehicleSpeed
Type	Integer
Parameter Value	2

Figure 5: Adding Monitoring Action

c. Adding Adjustment Action



Adjustment Action

Adjustment Action Name: VAA

Associated QoR Metric: vehicleAcr

Artifact

ArtifactID: Artifact_VAA

ArtifactDescription: bash script to deploy VAA

Location: http://128.130.172.215/j

Type: sh

RESTfulAPI: /VAA/rest/adjust

HttpMethod: PUT

Prerequisite Actions

Prerequisite Action 1: SAA

Adjustment Cases

Number of Parameters in Analytic Task: 1

Number of Parameters in Adjustment Case: 1

- Adjustment Case 1

- Estimated Result

  From: 90

  To: 100

- Analytic Task

  TaskName: kmeans

  Analytic Task Parameter

    Parameter 1: k

    Type: Integer

    Value: 5

- Adjustment Case Parameter

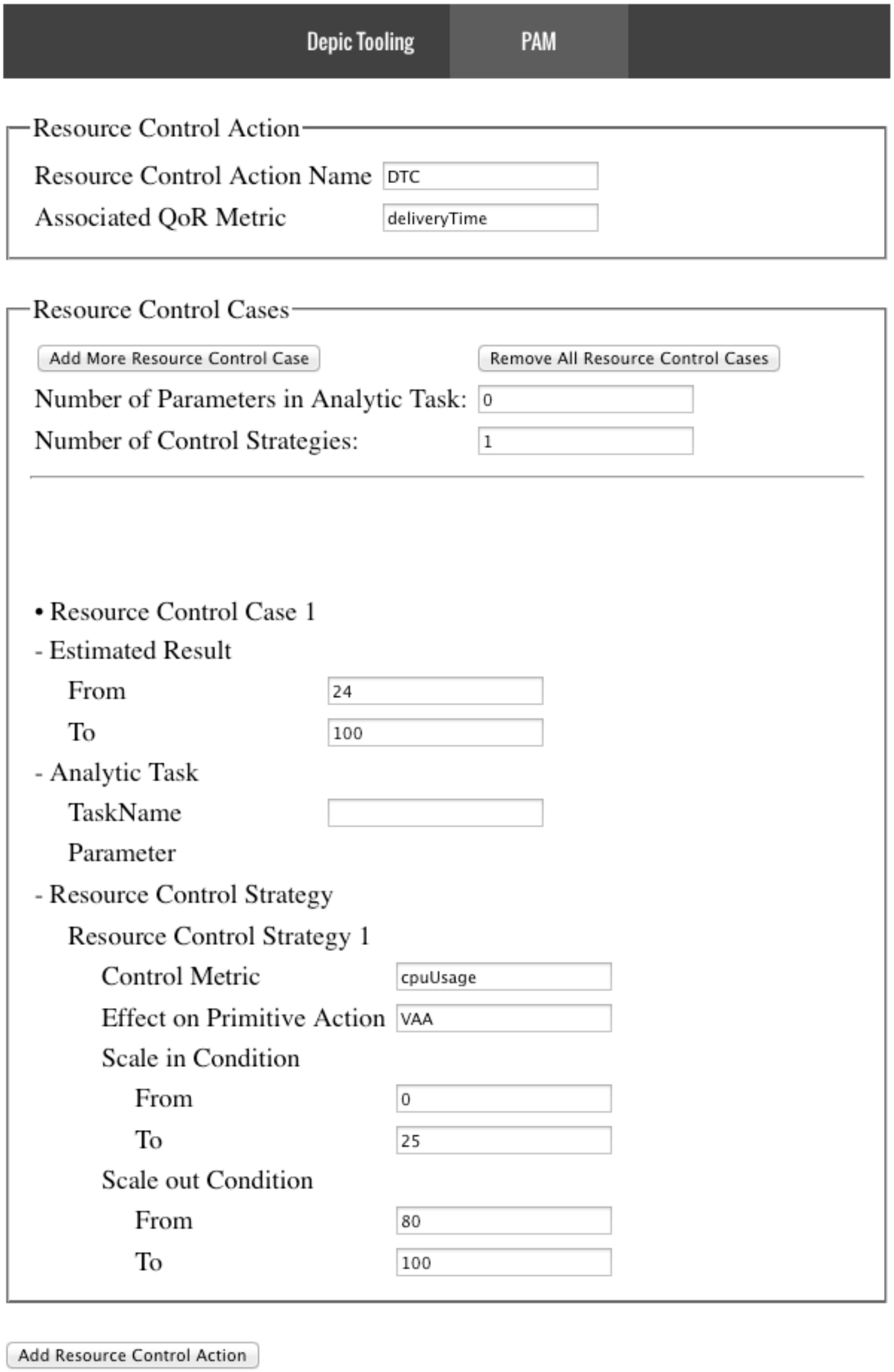
  Parameter 1: speedThreshold

  Type: Double

  Value: 10

Figure 6: Adding Adjustment Action

d. Adding Resource Control Action


Depic Tooling
PAM
Add More Resource Control Case
Remove All Resource Control Cases
Add Resource Control Action

Resource Control Action

Resource Control Action Name: DTC

Associated QoR Metric: deliveryTime

Resource Control Cases

Number of Parameters in Analytic Task: 0

Number of Control Strategies: 1

- Resource Control Case 1

- Estimated Result
  - From: 24
  - To: 100
- Analytic Task
  - TaskName:
  - Parameter
- Resource Control Strategy
  - Resource Control Strategy 1
    - Control Metric: cpuUsage
    - Effect on Primitive Action: VAA
    - Scale in Condition
      - From: 0
      - To: 25
    - Scale out Condition
      - From: 80
      - To: 100

Figure 7: Adding Resource Control Action