**DYFLOW XML:**

DYFLOW xml consists of three sections corresponding to Monitor, Decision and Arbitration stages of the dynamic management.

**MONITOR:**

Defines the sensors to use and tasks to apply these sensors on.

* Sensors are uniquely identified by sensor-ids and each sensor tracks a single variable (i.e., TAU generated or custom). Sensor type is used to define the stream/file format to read from.
  + The tag **group-by** could be used to define what post processing to perform on the sensor input. A post-processing group is defined by two attributes **granularity** and **reduction-operation** which defines the level at which the grouping should be performed and how the grouping would be done (i.e, SUM, MAX, MIN, AVG, STD, DIV, MUL, PERCENTAGE, FIRST, LAST, MODE. Multiple operations could be performed in a sequence, e.g. MIN MAX).  Users can define multiple groups for different granularities. There could be three levels of granularity; process-groups, node, task and ensemble.
    - TASK-NODE: The processes of the task are grouped based on the compute node they are assigned to.
    - TASK: The sensor output is considered based on the grouping of all the processes of the workflow task.
    - WORKFLOW-NODE: The processes of the monitored tasks of the workflow are grouped based on the compute node they are assigned to.
    - WORKFLOW: The sensor output is considered based on the grouping of all the processes of the monitored workflow tasks.
  + The tag **join** could be utilized to combine the current sensor output with other sensor outputs. E.g. IPC = (Instructions completed) / (Number of CPU cycles used)
* Once the sensors are defined, the users can separately define which tasks to apply these sensors on. Having a separate tag helps to support pre-defined sensors. Users need to provide the name of the task, stream name to use and sensor-ids to map to the stream. For each sensor-id, the variable to read and other sensor parameters can be specified.

**<monitor>**

**<!-- start of monitoring rules -->**

**<sensors>**

**<!— Encapsulates the configuration multiple sensors that will be use at runtime -->**

**<sensor id=”” type=”” >**

**<!— Configures a sensor that will read a single info/variable at runtime. The id attribute sets a unique user defined identity of a sensor. The type attribute defines the formats from which the data would be read at runtime. Possible values for type are ADIOS2, TAUADIOS2, DISKPOLL -->**

**<group-by>**

**<!— Configures different ways to group the data for this sensor -->**

**<group granularity=”” reduction-operation/>**

**<!— Each group will have a granularity level for grouping the data and a reduction-operation to use for grouping. Possible values for granularity levels are TASK-NODE, TASK, WORKFLOW-NODE, WORKFLOW. Possible values for reduction operation are MIN, MAX, MODE, FIRST, LAST, STD, AVG, SUM, PERCENTAGE, DIV -->**

**</group-by>**

**<join sensor-id=”” join-operation=”” />**

**<!-- Since a sensor tracks only a single variable, this tag gives users the facility to combine the output of any other sensor/variable to form a metric that depends on multiple variables. E.g. IPC = (Instructions completed) / (Number of CPU cycles used). Valid join operations could be SUM, AVG, DIV, MUL, PERCENTAGE -->**

**</sensor>**

**</sensors>**

**<monitor-tasks>**

**<!— This encapsulates the information about how the sensors will be applied to the workflow tasks -->**

**<monitor-task name=”” workflowId=”” info-source=””/>**

**<!-- Defines the tasks of the workflow would be**

**monitored and by which sensors. The attribute name**

**defines the task name, workflowId specifies the workflow**

**the task belongs to, and info-source defines the**

**stream/file to read data from -->**

**<use-sensor sensor-id=”” info=””>**

**<!-- Defines the sensor to apply to this task.**

**The attribute senor-id sets the sensor and info**

**defines the variable to read for sensor -->**

**<parameters>**

**<parameter key=”” value=””/>**

**</parameters>**

**<!-- Defines any sensor parameters. For e.g.,**

**in case of DISKPOLL, it is the frequency of**

**the file output and initial output. It can be**

**used in case of CUSTOM sensor operation. Each**

**parameter uses a key attribute to associate**

**the value -->**

**</use-sensor>**

**</monitor-task>**

**</monitor-tasks>**

**</monitor>**

**DECISION:**

Defines policies to use at runtime and how these policies will be applied to the workflow tasks.

* Each decision rule is uniquely identified and has a type. Each decision rule defines runtime triggers and action to perform if the trigger conditions are true
  + Each trigger needs a comparison operation (e.g. GT, LT, EQ, LEQ, GEQ, NEQ etc) and a threshold value.
  + The tag **use-sensor** uses **following attributes:** 
    - **id** attribute to define which sensor to use as the trigger input.
    - **granularity** defines the level of the sensor input to use. The rule will be applied to all the tasks that are monitored by this sensor.
    - **frequently** lets the user determine how often this rule should be invoked.
* The ***apply-policy*** tag is used to configure the policies for the workflow tasks.  The tag **apply-policy** is used to what policy is used for which workflow tasks. The tag **act-on-tasks** defines a set of the response of the policy will be applied to. The tag action-params is used to define any additional parameters to use for the policy actions.

.

**<-- start of decision rules -->**

**<decision>**

**<policies>**

**<-- Encapsulates the configuration of all decision policies set by -->**

**<policy id=”” >**

**<!— Each policy should have a unique user defined id 🡪**

 ­­­­­

**<eval operation=”” threshold =””/>**

**<-- Eval defines the runtime condition on the sensor output that if true will enable the decision. The operation could be GT, LT, EG, LEQ, GEQ, NEQ, RELATIVE. Threshold is a user provided double value. -->**

**<sensors-to-use>**

**<use-sensor id=”” granularity=”” />**

**<sensors-to-use>**

**<-- The sensor-to-use tag defines which sensor outputs to use. Each use-sensor tag defines a sensor to use and its granularity value. -->**

**<action></action>**

**<--The action tag defines what action to perform if the**

**runtime condition for the evaluation condition is true.**

**Possible values are ADDCPU, RMCPU, SWITCH, START, STOP -->**

**<history window=”” operation=”” />**

**<-- If the sensor values need to be observed and operated over a history. This tag can be used to set the length of a sliding window over sensor values and operation to perform over the history. The possible values of operation are MIN, MAX, AVG, STD -->**

**<frequency seconds=”” />**

**<-- The frequency can be used to set invoke the decision after specific intervals (in seconds) -->**

**</policy >**

**</policies>**

**<apply-on workflowId=””>**

**<--This tag encapsulates which policies will be applied to a workflow -->**

**<apply-policy policyId=”” assess-task-name=””>**

**<--The apply-policy tag defines which policy will be applied to a workflow task -->**

**<act-on-tasks> </act-on-tasks>**

**<action-params>**

**<param key=”” value=””/>**

**</action-params>**

**</apply-policy>**

**<!— This defines which tasks will be acted upon as the policy response for the monitored task. Additional parameters can be defined for the action -->**

**</apply-on>**

**</decision>**

**ARBITRATION:**

Defines rules to govern the suggested actions at runtime.

* The tag **rule-for** defines the arbitration rules for a workflow. The tags **task-priorities** and **task-dependencies** lets users prioritize tasks and determine task dependencies for this workflow.

**<arbitration>**

**<-- start of arbitration rules -->**

**<rules>**

**<-- Encapsulates all the rules -->**

**<rule-for workflowId=”” >**

**<-- Encapsulates all the rules for a workflow -->**

  **<task-priorities>**

**<task-priority name=”” priority=”” />**

**</task-priorities>**

**<-- Defines task priority-->**

**<task-dependencies>**

**<task-dep name=”” type=”” parent=”” />**

**</task-dependencies>**

**<-- Defines task dependencies -->**

**</rule-for>**

**<rules>**

**</arbitration>**