Last edited by **Ondřej Vašíček** 1 day ago

2. Analysis Tool Definition

The Universal Analysis Adapter can be configured to use any analysis tool needed by the user (or that is the goal at least). A tools configuration consists of two files - tool.rdf and tool.properties - placed in the "conf/analysis_advanced/AnalysisTools" directory. To add a new tool, create a new .rdf file and a .properties file with the same name. The .rdf file is to contain an Automation Plan rdf resource, and the .properties file is used to set other configuration for the tool. There is an example tool definition "ExampleTool" in the "tutorials/conf_example/analysis_advanced/AnalysisTools" directory.

The example .properties file looks like this (see comments for description):

```
## Path to the tool executable
## IMPORTANT: Use double backslash on windows! (\\ instead of just \)
toolLaunchCommand=/full/path/to/executable.sh

## Arguments to always use on the command line when launching the tool (e.g. to make the to
## adapter). These will always be placed as the first command line parameter.
#toolSpecificArgs=--example

## If set to true, then only one AutomationRequest executing this AutomationPlan will be ru
## remaining ones will be placed in a queue.
oneInstanceOnly=False
```

The example AutomationPlan definition looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:fit="http://fit.vutbr.cz/group/verifit/oslc/ns/universal-analysis#"
    xmlns:oslc_data="http://open-services.net/ns/servicemanagement/1.0/"
    xmlns:oslc="http://open-services.net/ns/core#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:oslc_auto="http://open-services.net/ns/auto#">

    <!--
        Use this AutomationPlan as an example for defining new ones.
    -->
    <oslc_auto:AutomationPlan>
        <dcterms:identifier>example</dcterms:identifier> <!-- What the last part of this Au

        <oslc_auto:usesExecutionEnvironment rdf:resource="https://url.to.your.tool.com"/> <
        <dcterms:title rdf:parseType="Literal">Example Tool</dcterms:title>   <!-- Non-func
        <dcterms:description rdf:parseType="Literal">Used as an example.</dcterms:descripti
        <dcterms:creator rdf:resource="https://url.to.the.creator.com"/> <!-- Non-functiona

        <!--
            Parameter definitions start here. Each parameter definition defines a possible
            Use them to define the commandline interface of your tool. When executing an Au
            on the specified commandline position (starting at 1).

            The adapter will add a number of other ParameterDefinitions to the AutoPlan on
        -->

        <oslc_auto:parameterDefinition>
            <!--
                This ParameterDefintion stands for inserting any string as the tools argume
                can be defined as a separate ParameterDefinition. That would allow the Auto
                always possible (eg. for complex commanline interfaces).
            -->
            <oslc_auto:ParameterDefinition>
                <oslc:name>arguments</oslc:name>
                <fit:commandlinePosition rdf:datatype="http://www.w3.org/2001/XMLSchema#int
                <oslc:occurs rdf:resource="http://open-services.net/ns/core#Exactly-one"/>
                <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
                <dcterms:description rdf:parseType="Literal">Specify which arguments should
            </oslc_auto:ParameterDefinition>
```

```
            </oslc_auto:parameterDefinition>

            <oslc_auto:parameterDefinition>
                <!--
                    launchSUT is a special ParameterDefinition recognized by the adapter. The a
                    parameter if this parameters value is true.
                -->
                <oslc_auto:ParameterDefinition>
                    <oslc:name>launchSUT</oslc:name>
                    <fit:commandlinePosition rdf:datatype="http://www.w3.org/2001/XMLSchema#int
                    <oslc:defaultValue>True</oslc:defaultValue>
                    <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-One"/>
                    <dcterms:description rdf:parseType="Literal">This parameter definitions tel
                </oslc_auto:ParameterDefinition>
            </oslc_auto:parameterDefinition>

            <oslc_auto:parameterDefinition>
                <!--
                    SUTbuildCommand is the same as "launchSUT" except that the SUT build comman
                -->
                <oslc_auto:ParameterDefinition>
                    <oslc:name>SUTbuildCommand</oslc:name>
                    <fit:commandlinePosition rdf:datatype="http://www.w3.org/2001/XMLSchema#int
                    <oslc:defaultValue>True</oslc:defaultValue>
                    <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-One"/>
                    <dcterms:description rdf:parseType="Literal">This parameter definitions tel
                </oslc_auto:ParameterDefinition>
            </oslc_auto:parameterDefinition>

        </oslc_auto:AutomationPlan>

    </rdf:RDF>
```

The core functional properties are:

- dcterms:identifier - Determines the AutomationPlan's ID that will be used as the end of its URI. Has to be unique among all tools defined in an adapter.
- oslc_auto:parameterDefinition - Define the command line interface for the analysis tool.

Parameter Definitions are defined based on the desired configuration for that specific tool. Note that the adapter will add additional parameter definitions that are common for all AutomationPlans such as an SUT reference, timeout, output regex, and a zip flag. There are two types of parameter definitions a user can define for a tool.

- regular command line interface - These parameter definitions contain the *fit:commandlinePosition* property (see example above) which tells the adapter to place the value of the corresponding input parameter to that position on the command line when executing the tool. Any number of parameter definitions can be used. Command line positions need to be values from <1,inf>.
    - Useful properties of a parameter definition:
        - name - Used to match the parameter definition with input parameters used with an AutomationRequest
        - commandlinePosition - Determines where to place this paramter's value on the command line. Parameters with the same position will have random ordering relative to each other.
        - defaultValue - This value will be used when a corresponding input parameter is not supplied.
        - occurs - Defines whether the parameter is optional, or required. If a required parameter is not supplied with an AutomationRequest and there is no defaultValue defined, then the AutomationRequest creation will return an error.
        - allowedValue - Defines a single value that is allowed for this parameter's value. Use multiple times to define multiple values. If this property is defined for an AutomationPlan and an invalid value is supplied with a AutomationRequest, then the AutomationRequest creation will return an error.
        - valuePrefix - Defines a string to always be added as a prefix when placing values of this parameter definition on the command line.
        - valueType - Information for clients on what kind of value does this parameter definition expect.
        - readonly / hidden - Information for clients on whether this parameter is meant for them to see and use.
- Overriding default values of common parameter definitions (e.g. timeout, outputFileRegex, ...)
    - Defining a parameter definition with a defaultValue and the same name as one of the common parameter definitions will cause the defaultValue of the common parameter definition to be overwritten by the custom one.
- Special parameter definitions - Optional parameters definitions recognized by the analysis adapter with special functionality. These also have a commandlinePosition
    - launchSUT - This parameter's value toggles placing the SUT launchCommand being inserted at a given commandline position. The Analysis adapter can find the launchCommand by getting the SUT resource from the Compilation adapter (useful for dynamic analysis).

- SUTbuildCommand - Similar to the launchSUT parameter, only looks up and inserts the SUT buildCommand instead of the launchCommand (useful for static analysis).