GRANT AGREEMENT: 601138 | SCHEME FP7 ICT 2011.4.3
Promoting and Enhancing Reuse of Information throughout the Content Lifecycle taking account of Evolving Semantics [Digital Preservation]

Policy Editor

By Space Applications Services
(david.deweerdt@spaceapplications.com)

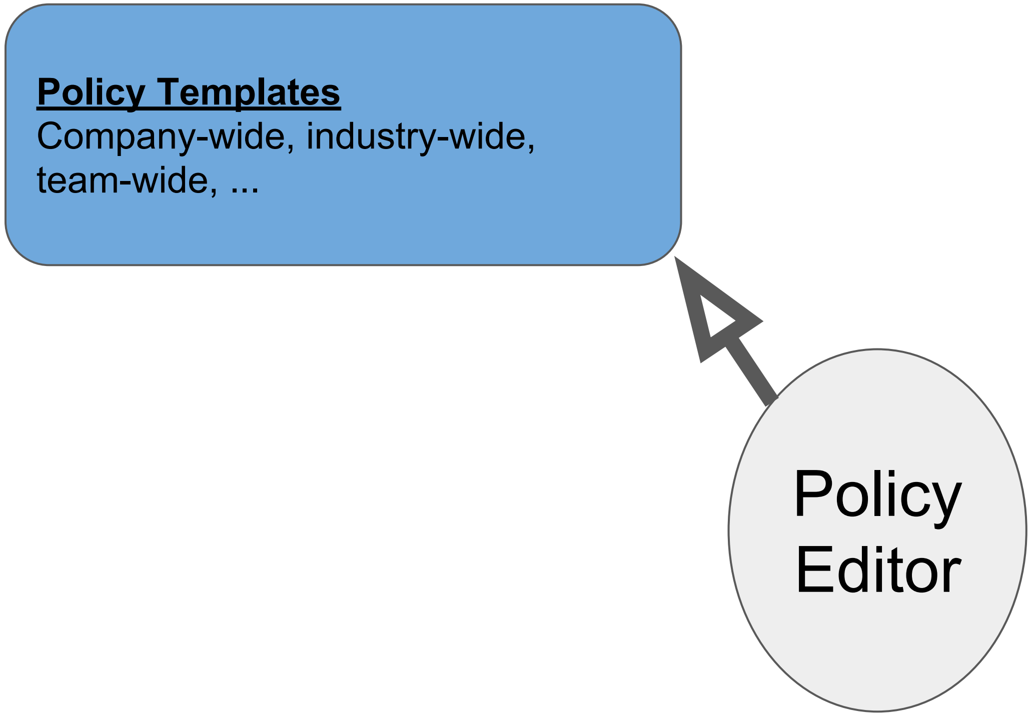
"This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no601138".

Policy Editor – What Is It?

- Proprietary web-based tool
- It facilitates the creation, modification and, optionally, verification and execution of a set of policies and procedures.
- It features a template- and variables-based approach allowing an a priori definition of domain- or organisational unit-specific policy blueprints, called policy templates. These templates are mixed and matched into concrete application-specific policies.
- It allows for implicit and automatic consistency and completeness checks and constraints enforcement.
- The Policy Editor can be instantiated with a multi-layered configurable policy model.
- Integration with Digital Ecosystem Models allows ensuring consistency between application policies and the ecosystem that these policies apply to.
- Optional support for executable low-level policies (called "processes") allowing for example dependency monitoring, Digital Ecosystem Model updating, notifications and action triggering.
- Export abilities to JSON and a printable PDF format.
- Potential applications: Automated Data Management Plans, small & large-scale QA definition, automatic constraint verification and enforcing.

Policy Editor - Main Features

Policy Editor - Main Features



Policy Editor - Main Features

Policy Templates
Company-wide, industry-wide, team-wide, ...

Organization specific policy template

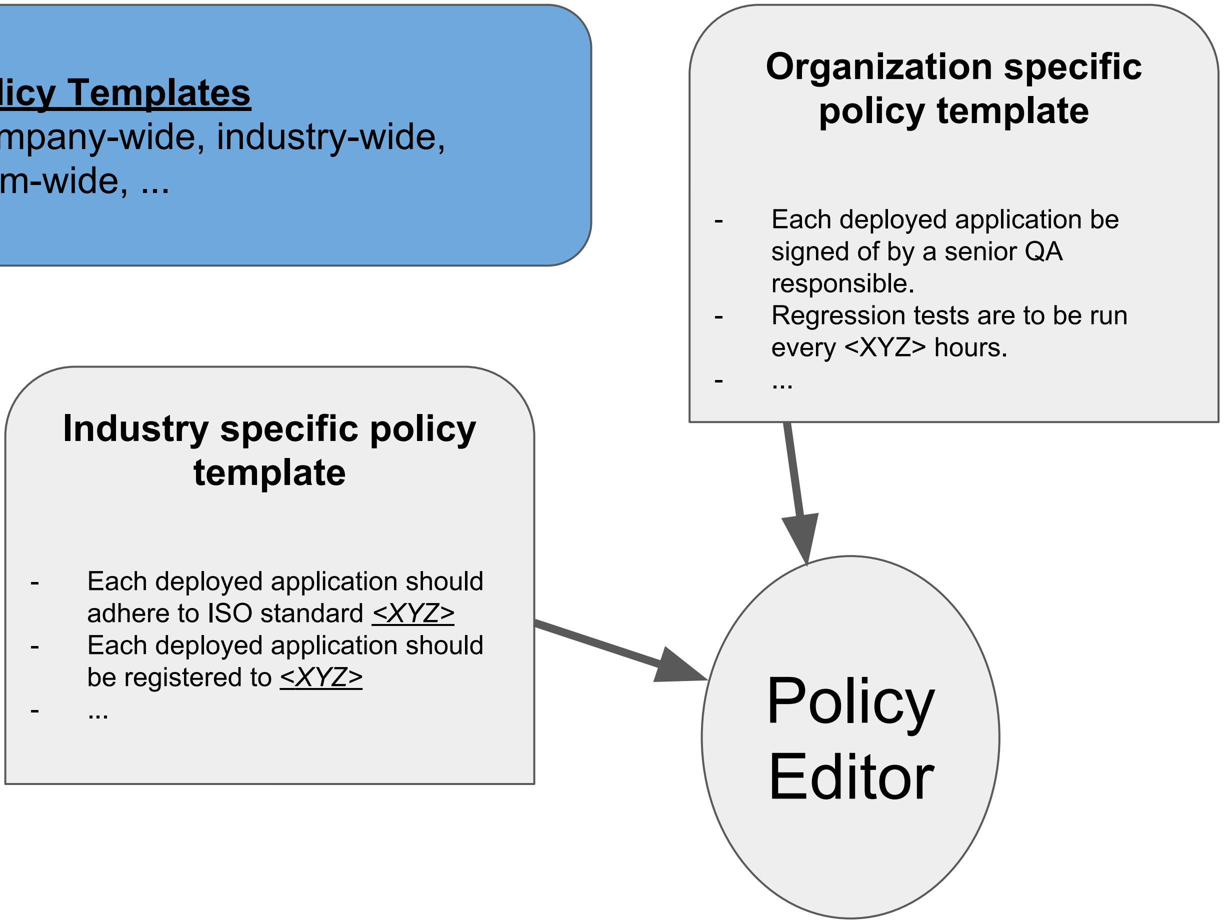
- Each deployed application be signed of by a senior QA responsible.
- Regression tests are to be run every <XYZ> hours.
- ...

Industry specific policy template

- Each deployed application should adhere to ISO standard *<XYZ>*
- Each deployed application should be registered to *<XYZ>*
- ...

Policy Editor - Main Features

Policy Templates
Company-wide, industry-wide, team-wide, ...



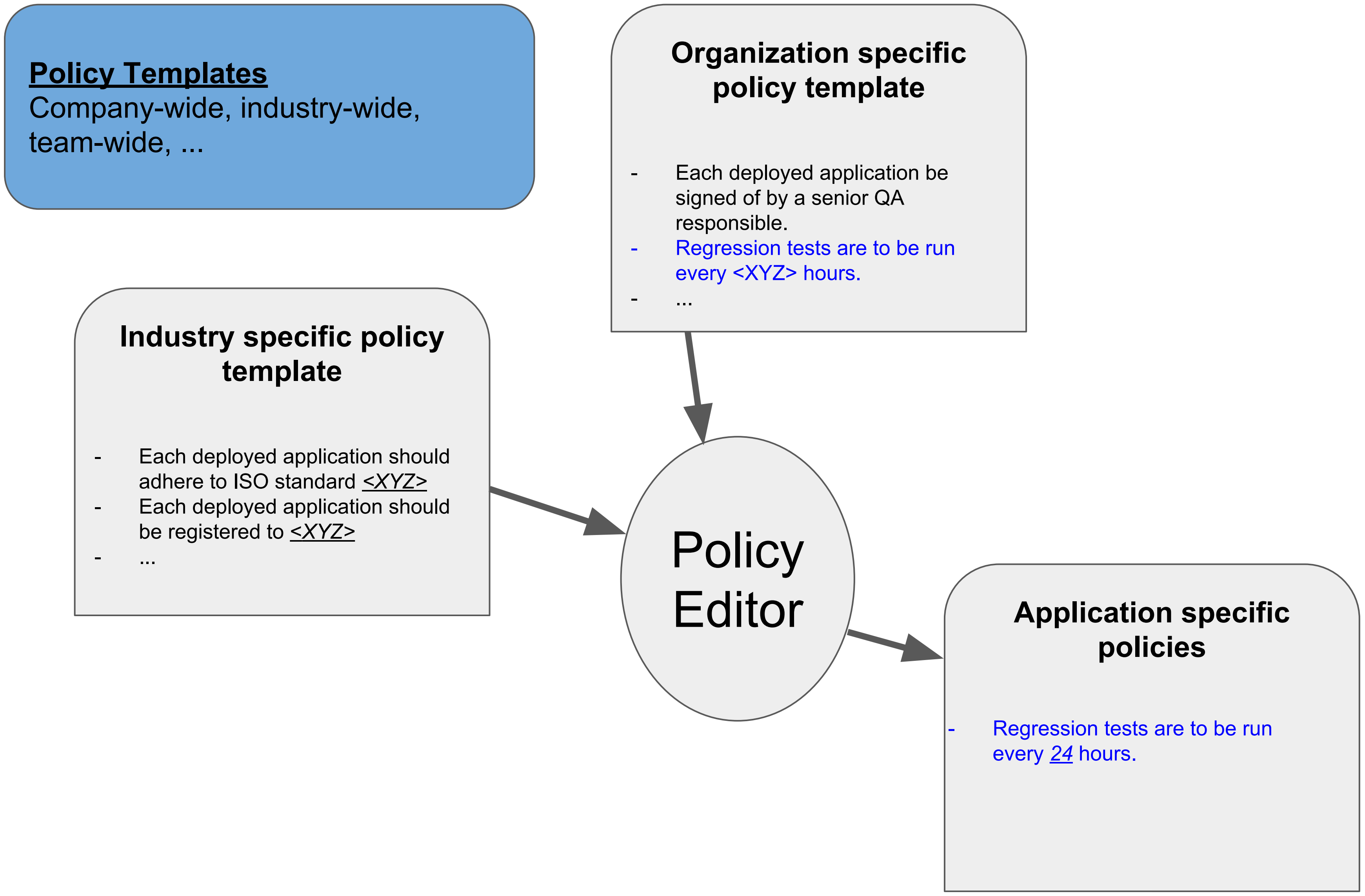
Organization specific policy template

- Each deployed application be signed of by a senior QA responsible.
- Regression tests are to be run every <XYZ> hours.
- ...

Industry specific policy template

- Each deployed application should adhere to ISO standard *<XYZ>*
- Each deployed application should be registered to *<XYZ>*
- ...

Policy Editor - Main Features



Policy Templates
Company-wide, industry-wide, team-wide, ...

Organization specific policy template

- Each deployed application be signed of by a senior QA responsible.
- Regression tests are to be run every <XYZ> hours.
- ...

Industry specific policy template

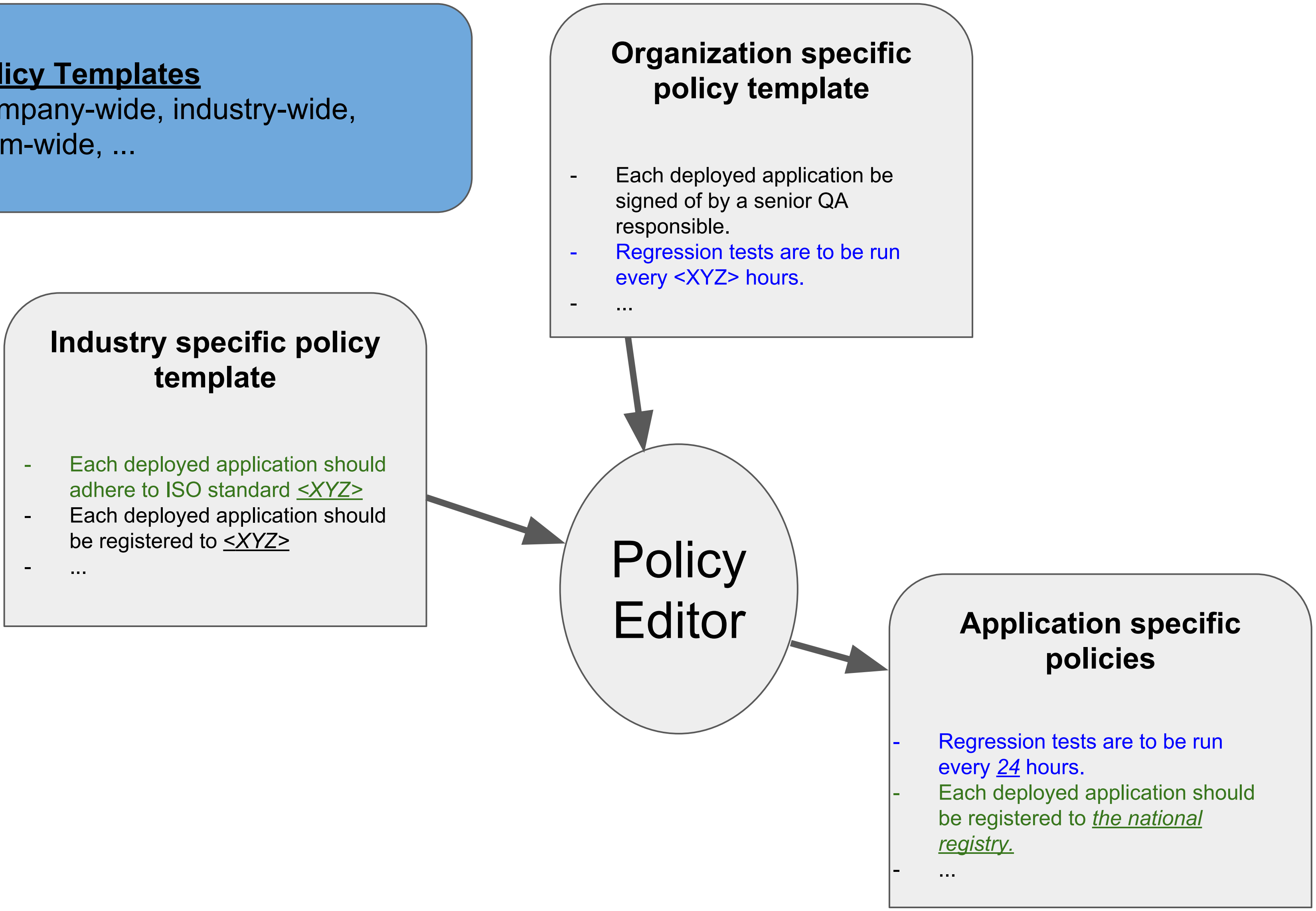
- Each deployed application should adhere to ISO standard *<XYZ>*
- Each deployed application should be registered to *<XYZ>*
- ...

Application specific policies

- Regression tests are to be run every *24* hours.

Policy Editor - Main Features

Policy Templates
Company-wide, industry-wide, team-wide, ...



Organization specific policy template

- Each deployed application be signed of by a senior QA responsible.
- Regression tests are to be run every <XYZ> hours.
- ...

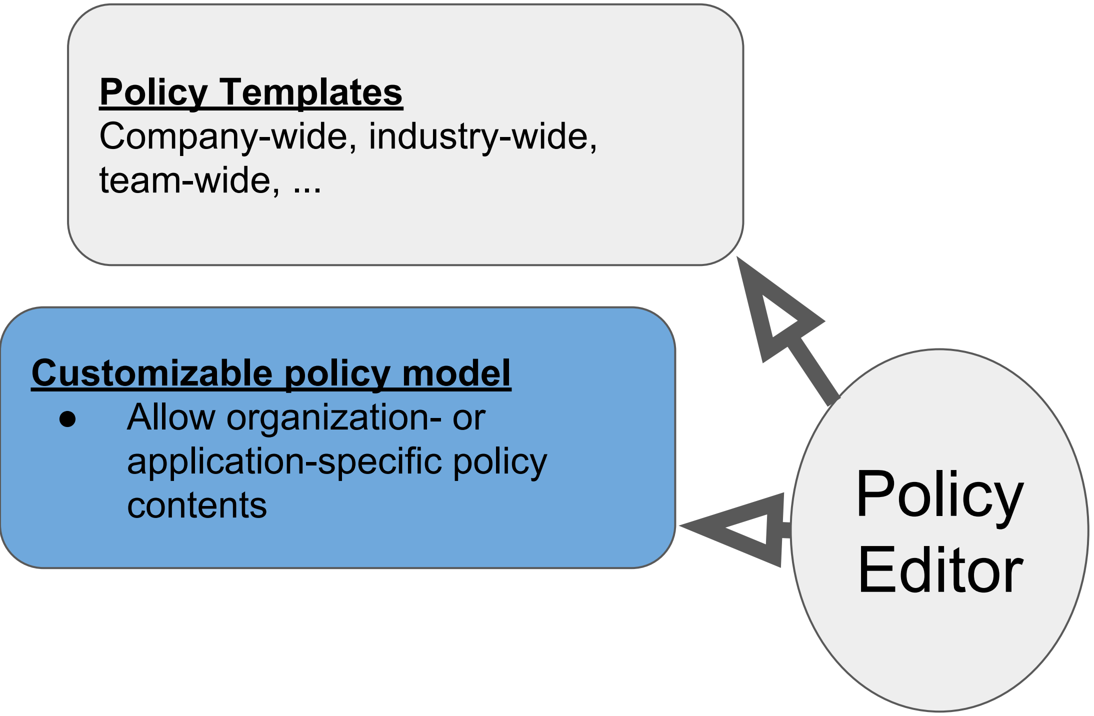
Industry specific policy template

- Each deployed application should adhere to ISO standard *<XYZ>*
- Each deployed application should be registered to *<XYZ>*
- ...

Application specific policies

- Regression tests are to be run every *24* hours.
- Each deployed application should be registered to *the national registry.*
- ...

Policy Editor - Main Features



<u>Policy Templates</u>
Company-wide, industry-wide, team-wide, ...

<u>Customizable policy model</u>
- Allow organization- or application-specific policy contents

Policy Editor - Main Features

Simple Policy Model

- Policy Text
- Author
- Version

Complex Policy Model

- Policy Text
- Author
- Version
- Maintainer
- Constraints
- Applicability
- Format
- Language
- Compliance
- Target users
- Replaced policies

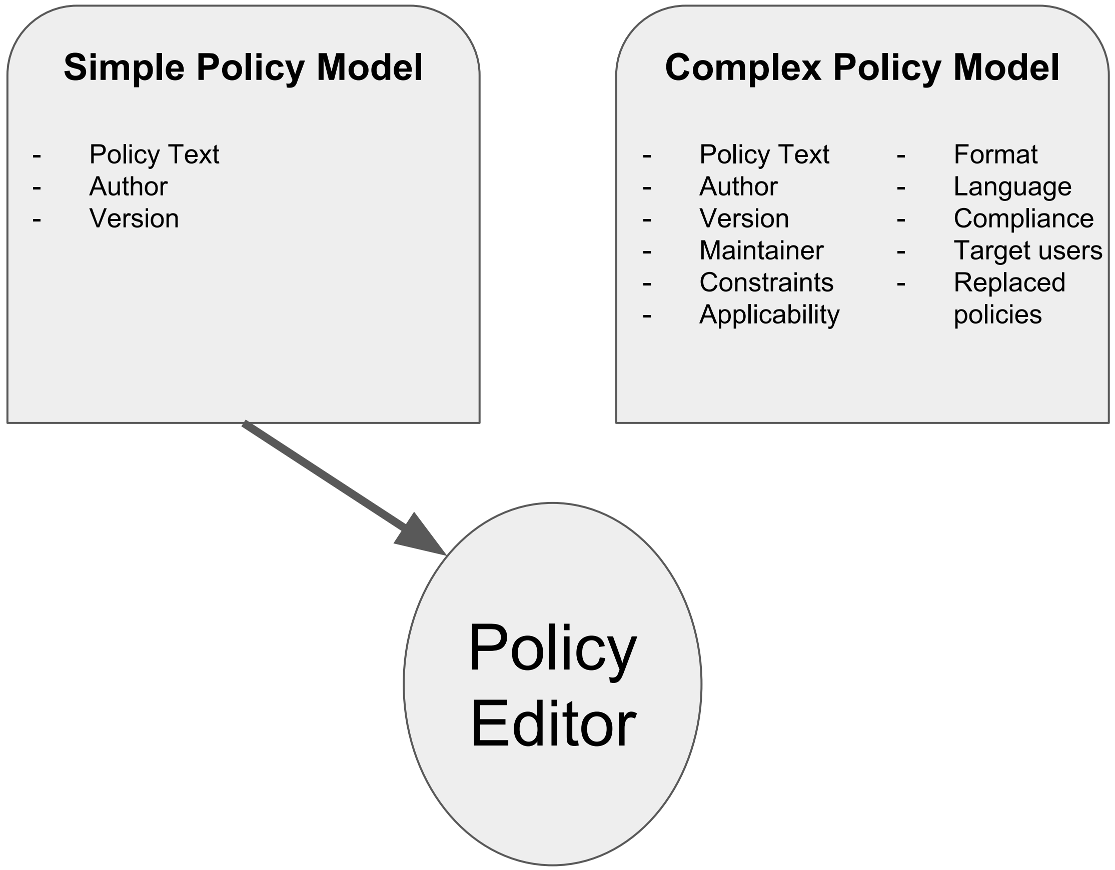
Customizable policy model

- Allow organization- or application-specific policy contents

Policy Editor - Main Features

Customizable policy model

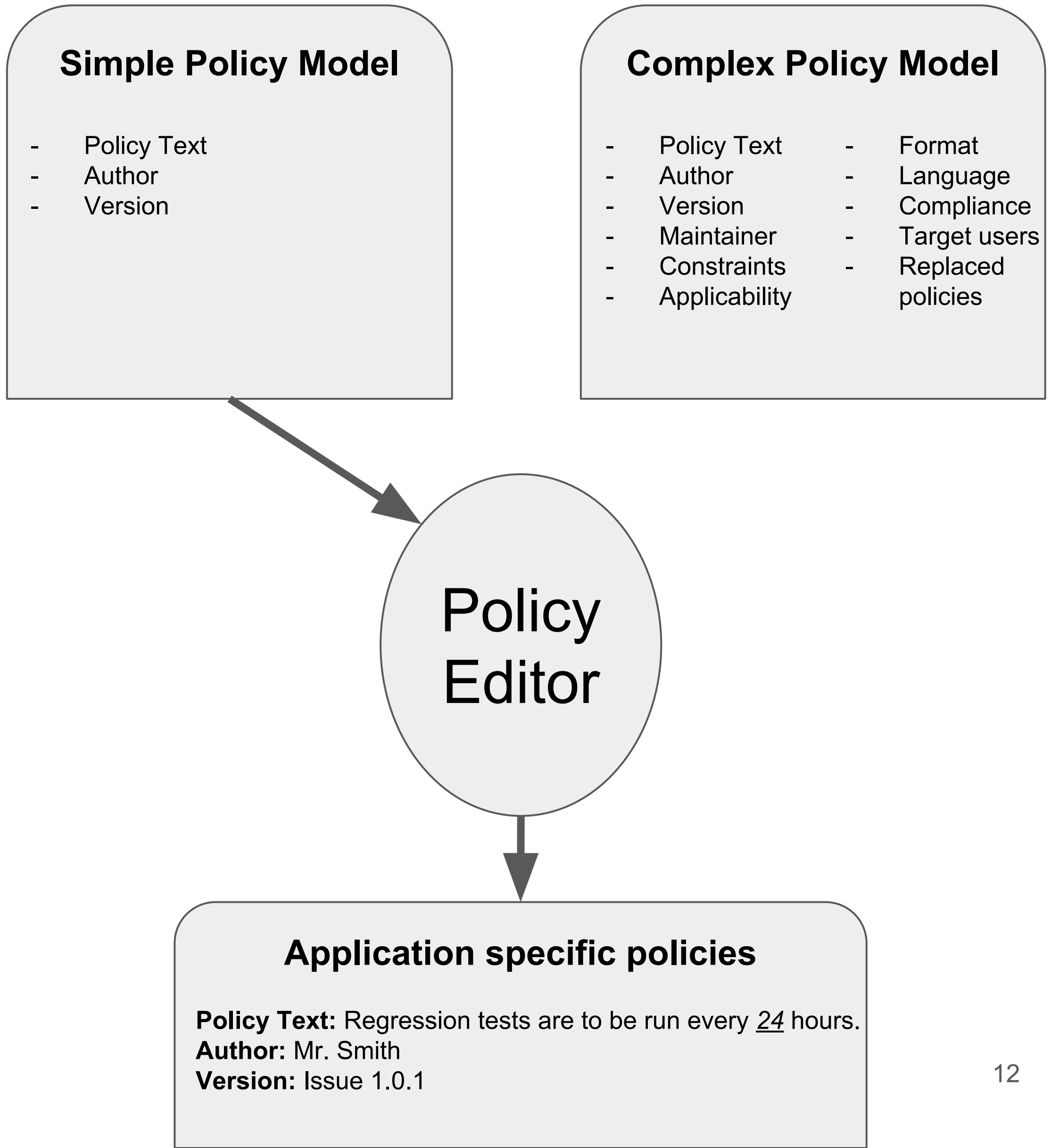
- Allow organization- or application-specific policy contents



Policy Editor - Main Features

Customizable policy model

- Allow organization- or application-specific policy contents



Simple Policy Model

- Policy Text
- Author
- Version

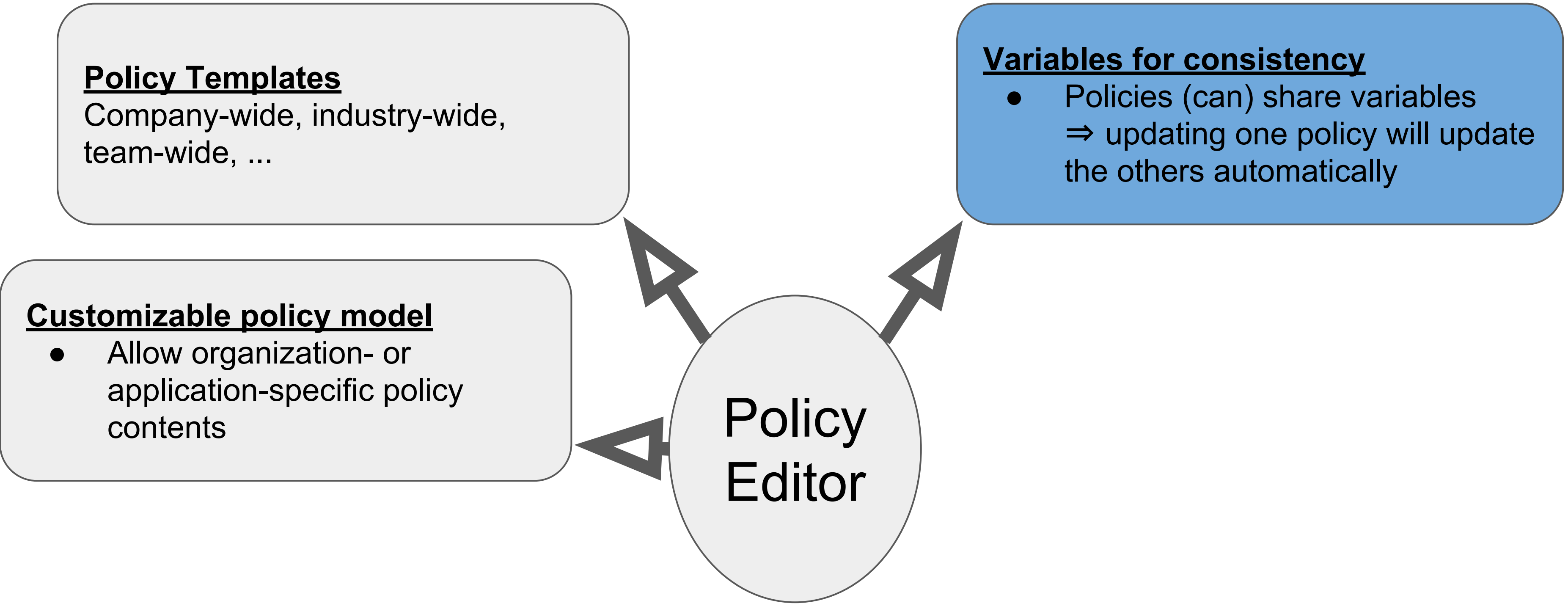
Complex Policy Model

- Policy Text
- Author
- Version
- Maintainer
- Constraints
- Applicability
- Format
- Language
- Compliance
- Target users
- Replaced policies

Application specific policies

Policy Text: Regression tests are to be run every _24_ hours.
Author: Mr. Smith
Version: Issue 1.0.1

Policy Editor - Main Features



Policy Templates
Company-wide, industry-wide, team-wide, ...

Variables for consistency
- Policies (can) share variables ⇒ updating one policy will update the others automatically

Customizable policy model
- Allow organization- or application-specific policy contents

Policy Editor - Main Features

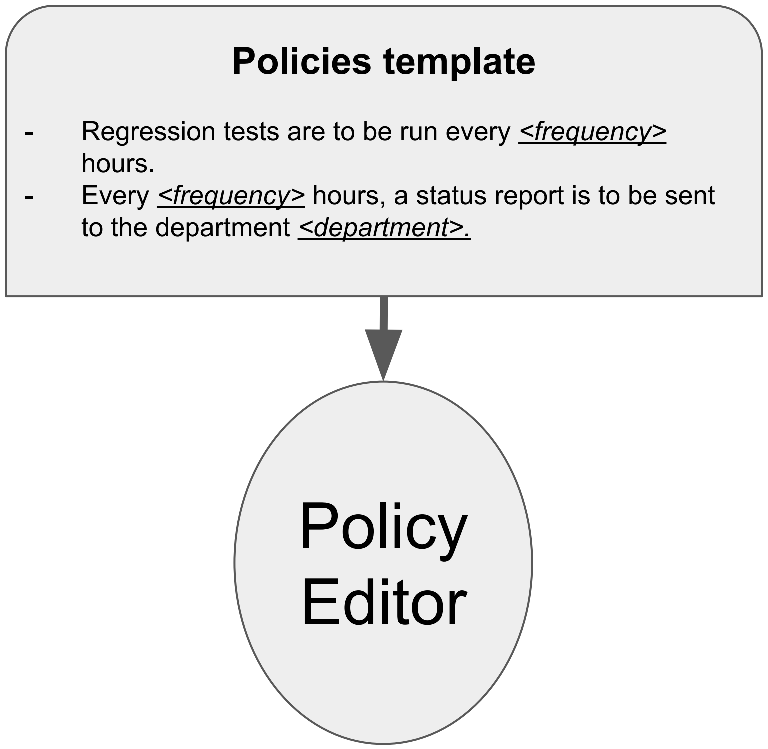
Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

Variables for consistency

- Policies (can) share variables
  ⇒ updating one policy will update the others automatically

Policy Editor - Main Features



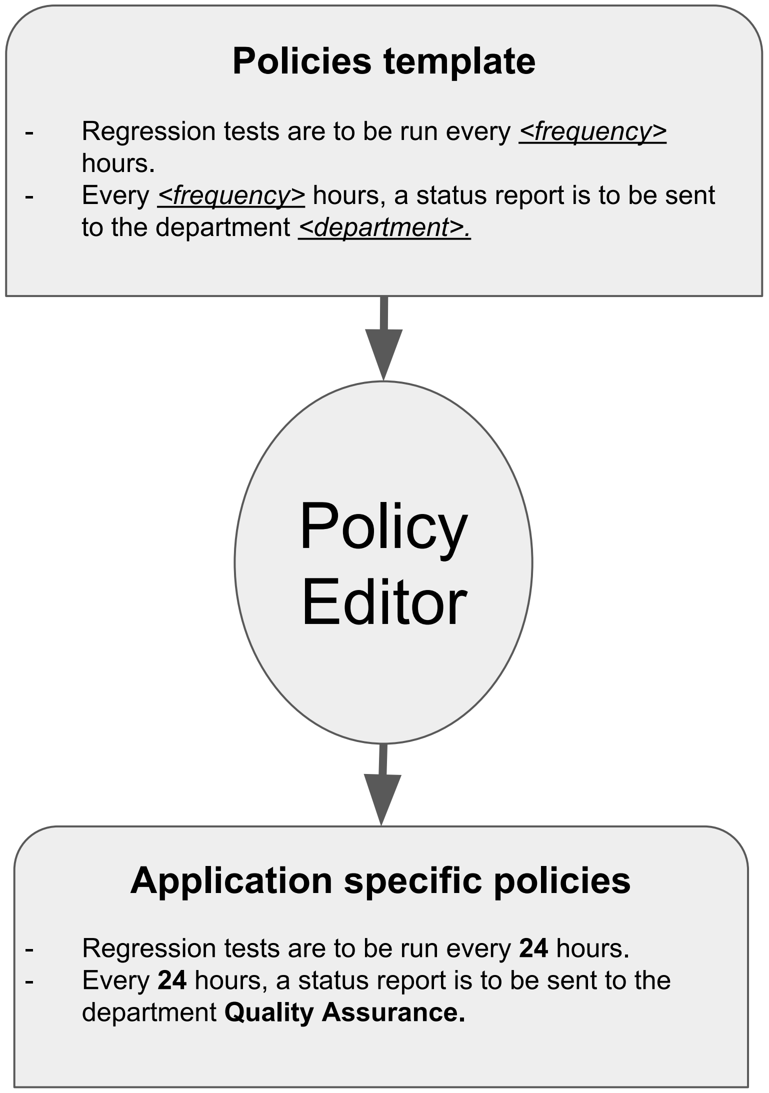
Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

Variables for consistency

- Policies (can) share variables ⇒ updating one policy will update the others automatically

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

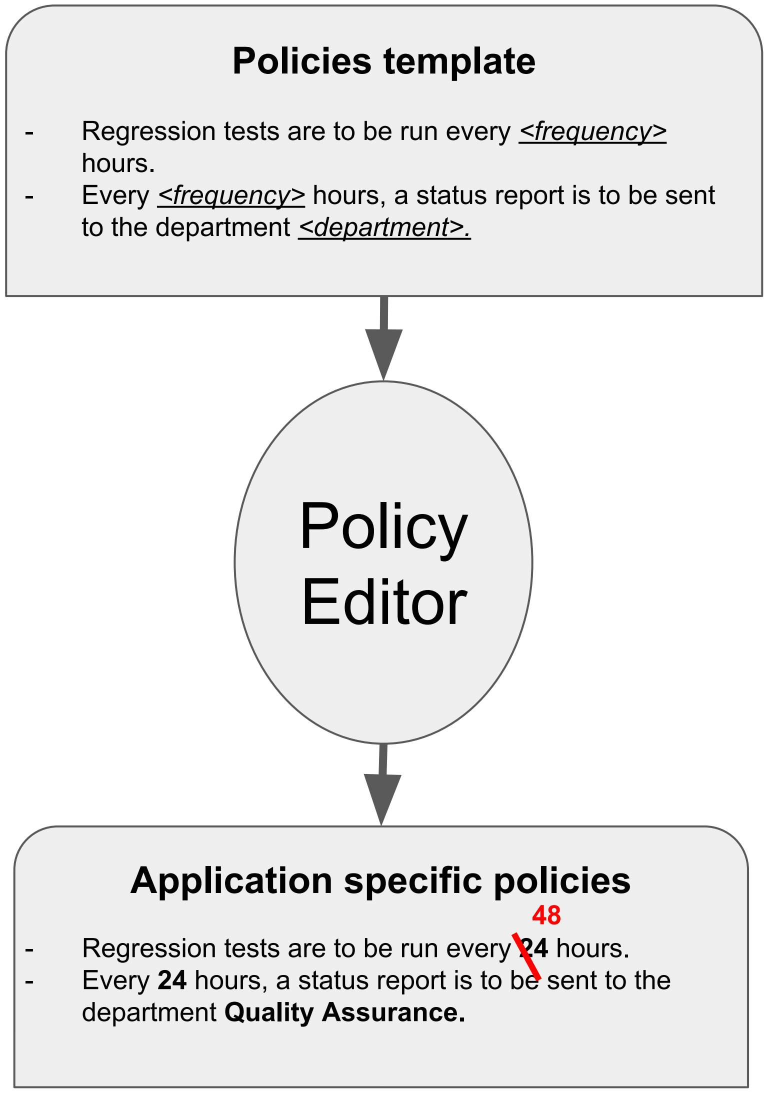
Application specific policies

- Regression tests are to be run every **24** hours.
- Every **24** hours, a status report is to be sent to the department **Quality Assurance.**

Variables for consistency

- Policies (can) share variables
  ⇒ updating one policy will update the others automatically

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

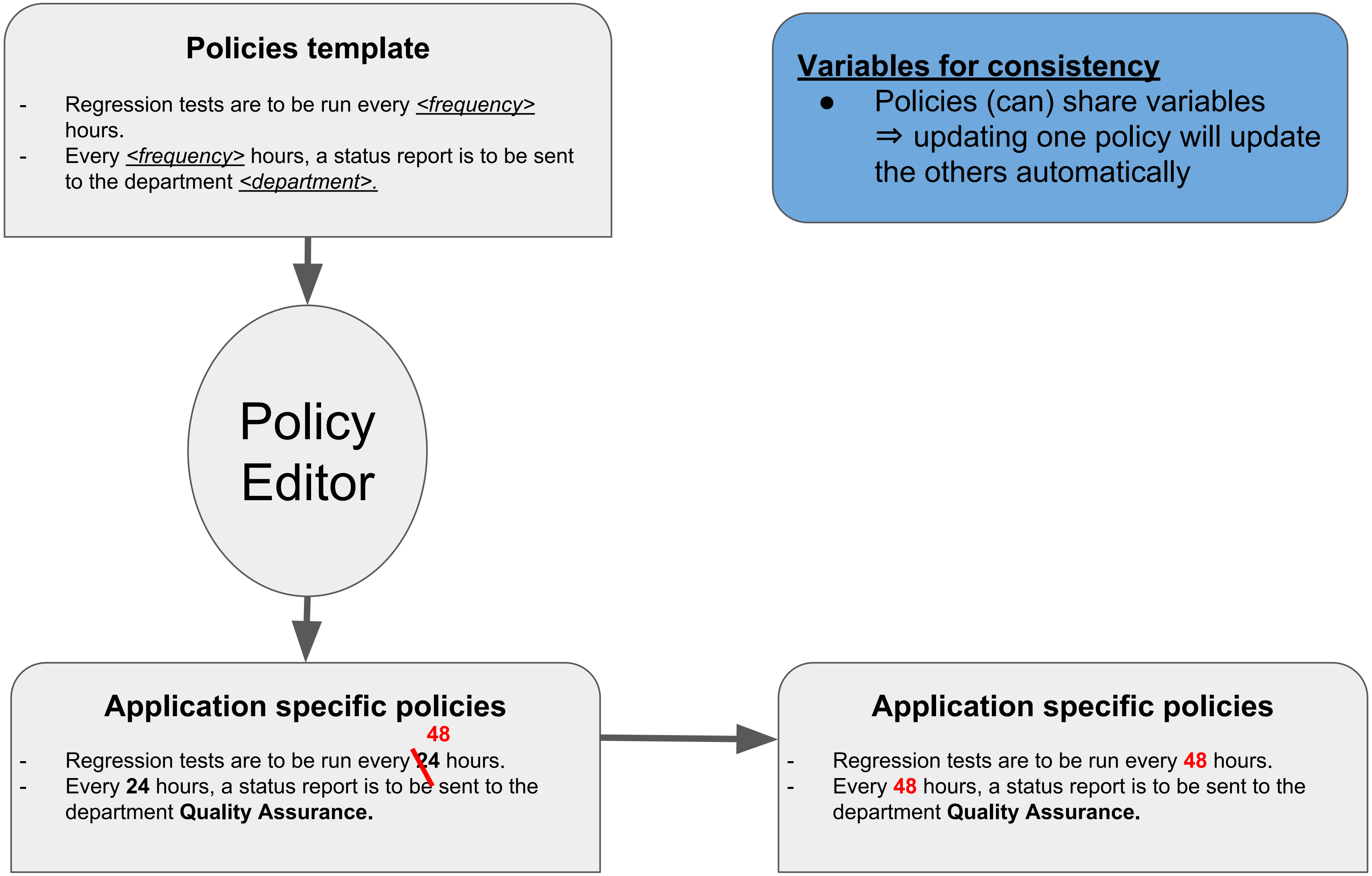
Application specific policies

- Regression tests are to be run every ~~**24**~~ **48** hours.
- Every **24** hours, a status report is to be sent to the department **Quality Assurance.**

Variables for consistency

- Policies (can) share variables ⇒ updating one policy will update the others automatically

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*

Application specific policies

- Regression tests are to be run every ~~**24**~~ **48** hours.
- Every **24** hours, a status report is to be sent to the department **Quality Assurance.**

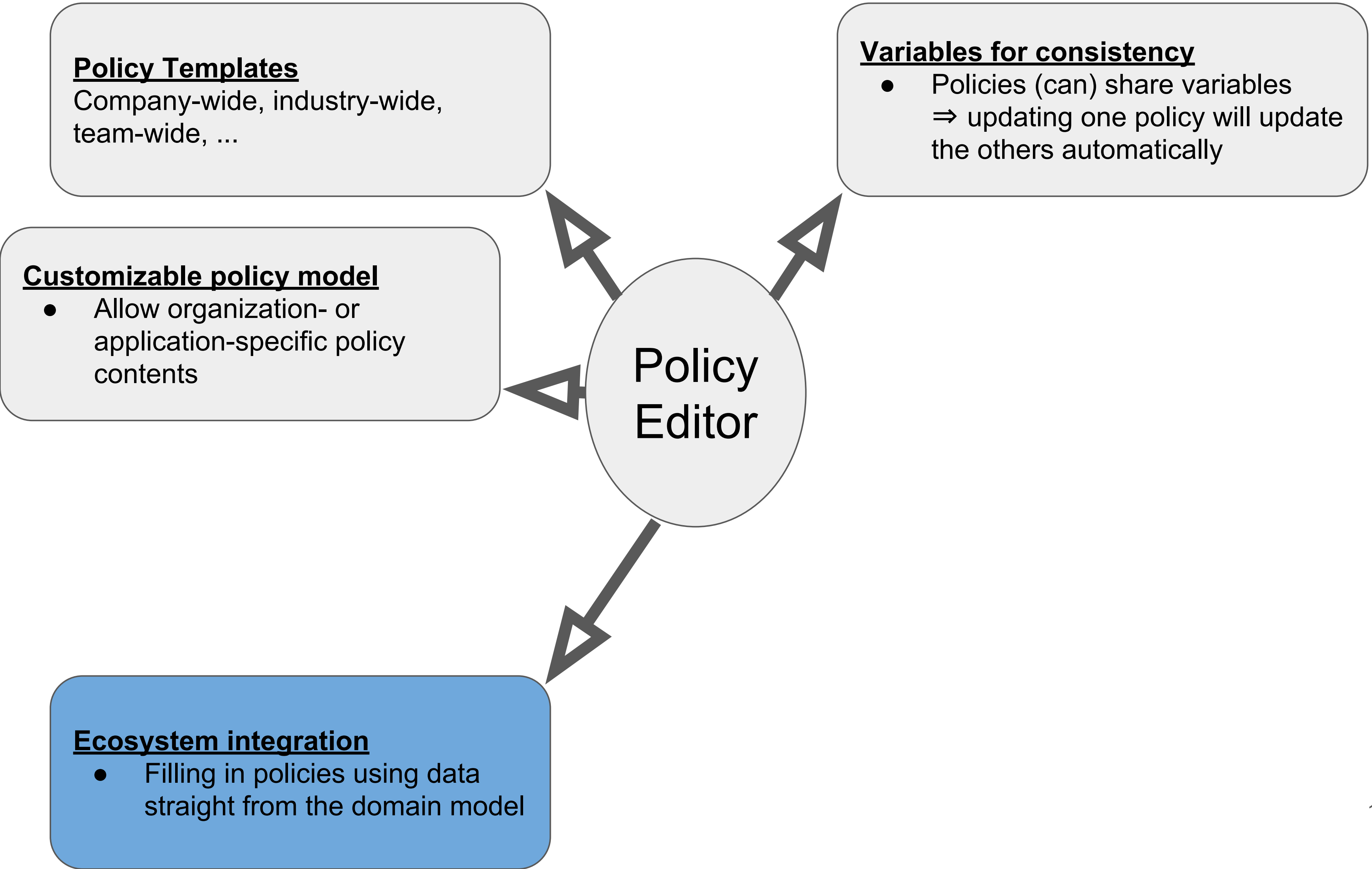
Application specific policies

- Regression tests are to be run every **48** hours.
- Every **48** hours, a status report is to be sent to the department **Quality Assurance.**

<u>Variables for consistency</u>

- Policies (can) share variables ⇒ updating one policy will update the others automatically

Policy Editor - Main Features



Policy Templates
Company-wide, industry-wide, team-wide, ...

Variables for consistency
- Policies (can) share variables ⇒ updating one policy will update the others automatically

Customizable policy model
- Allow organization- or application-specific policy contents

Ecosystem integration
- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*

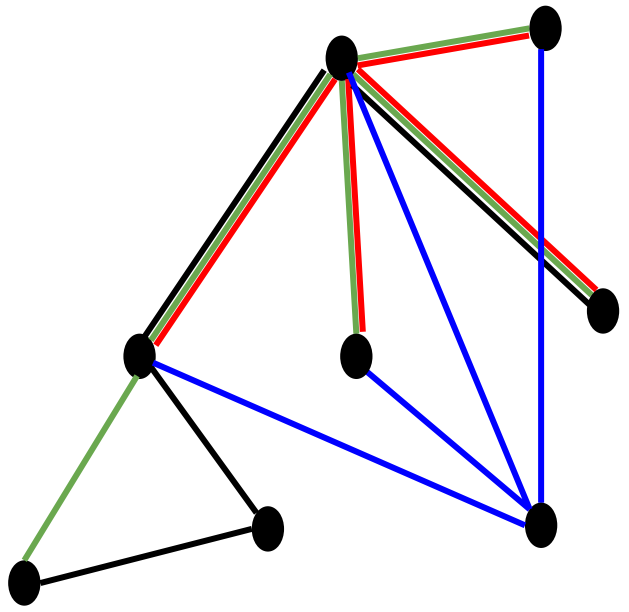
<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

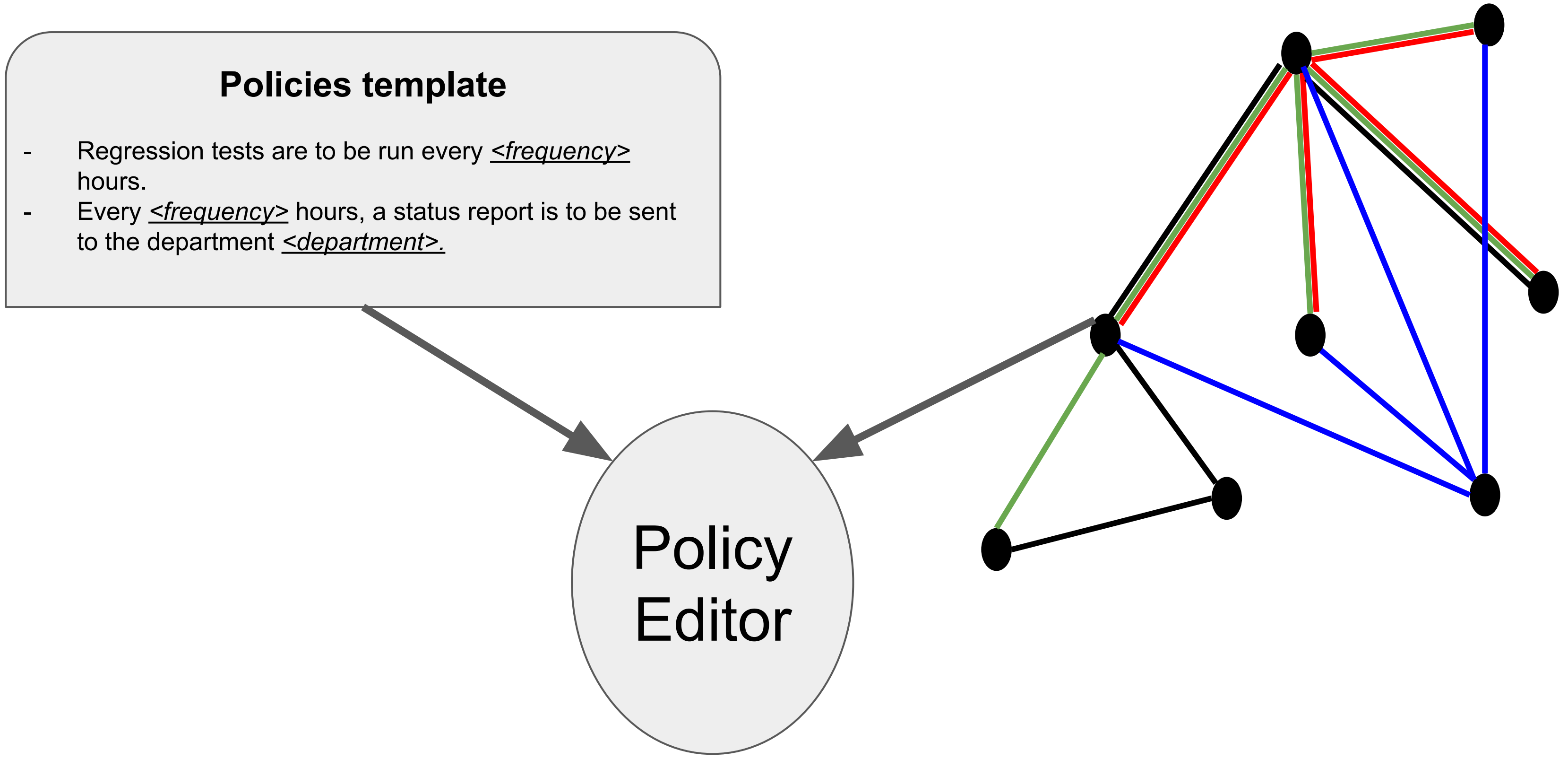
- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*



Ecosystem integration

- Filling in policies using data straight from the domain model

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*

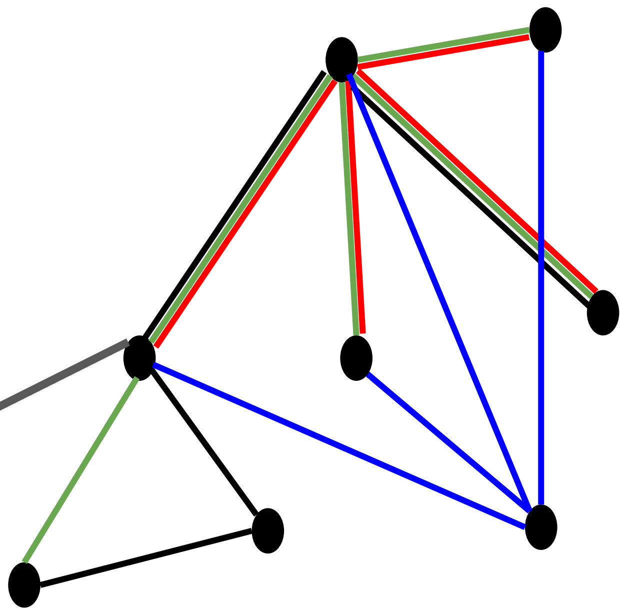
<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*



Policy Editor

Application specific policies

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department></u>*

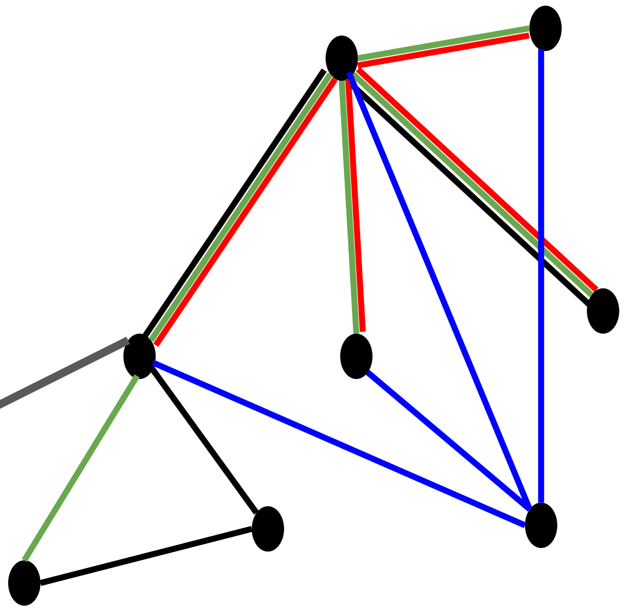
<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*



Policy Editor

Application specific policies

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department></u>*

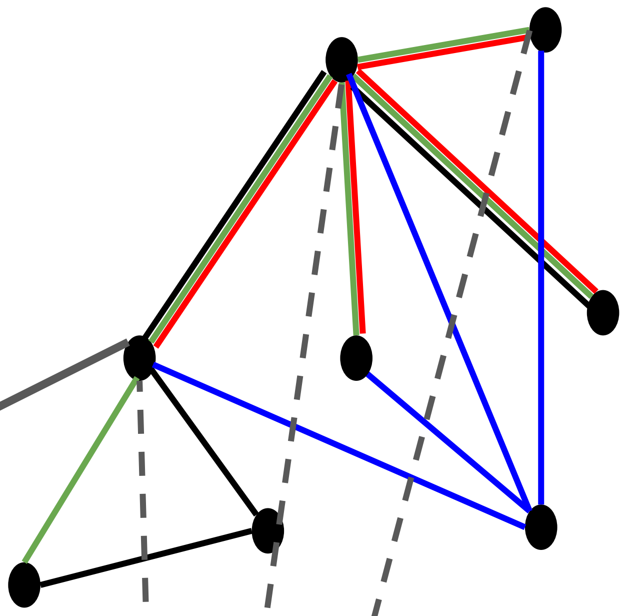
<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*



Policy Editor

Application specific policies

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the departme

Management department
QA department
R&D department

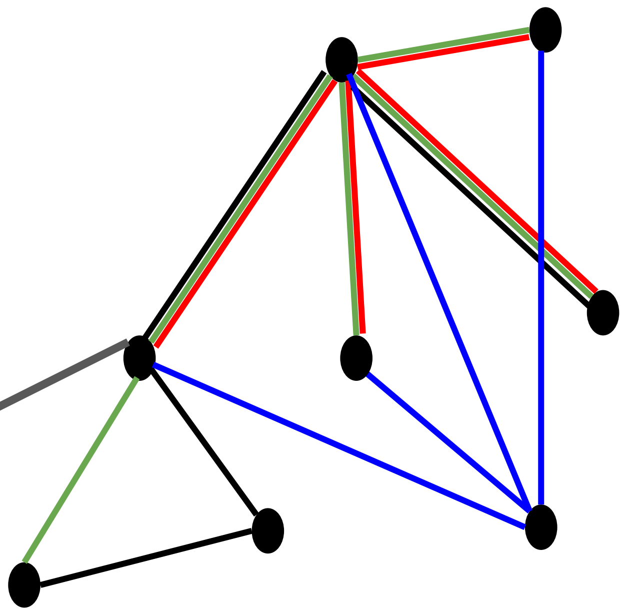
<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features

Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*



Policy Editor

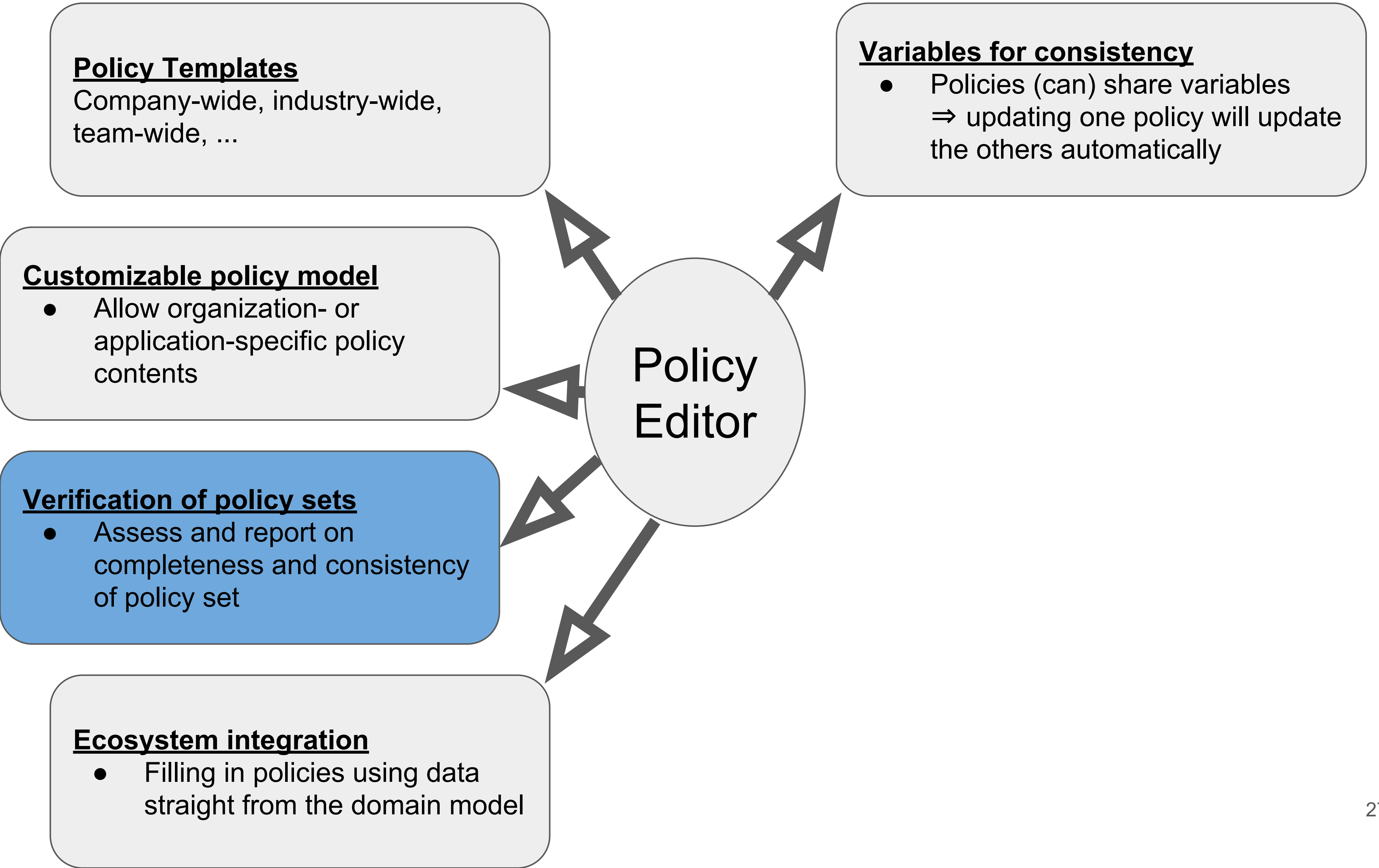
Application specific policies

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department **QA Department**

<u>Ecosystem integration</u>

- Filling in policies using data straight from the domain model

Policy Editor - Main Features



Policy Templates
Company-wide, industry-wide, team-wide, ...

Variables for consistency

- Policies (can) share variables ⇒ updating one policy will update the others automatically

Customizable policy model

- Allow organization- or application-specific policy contents

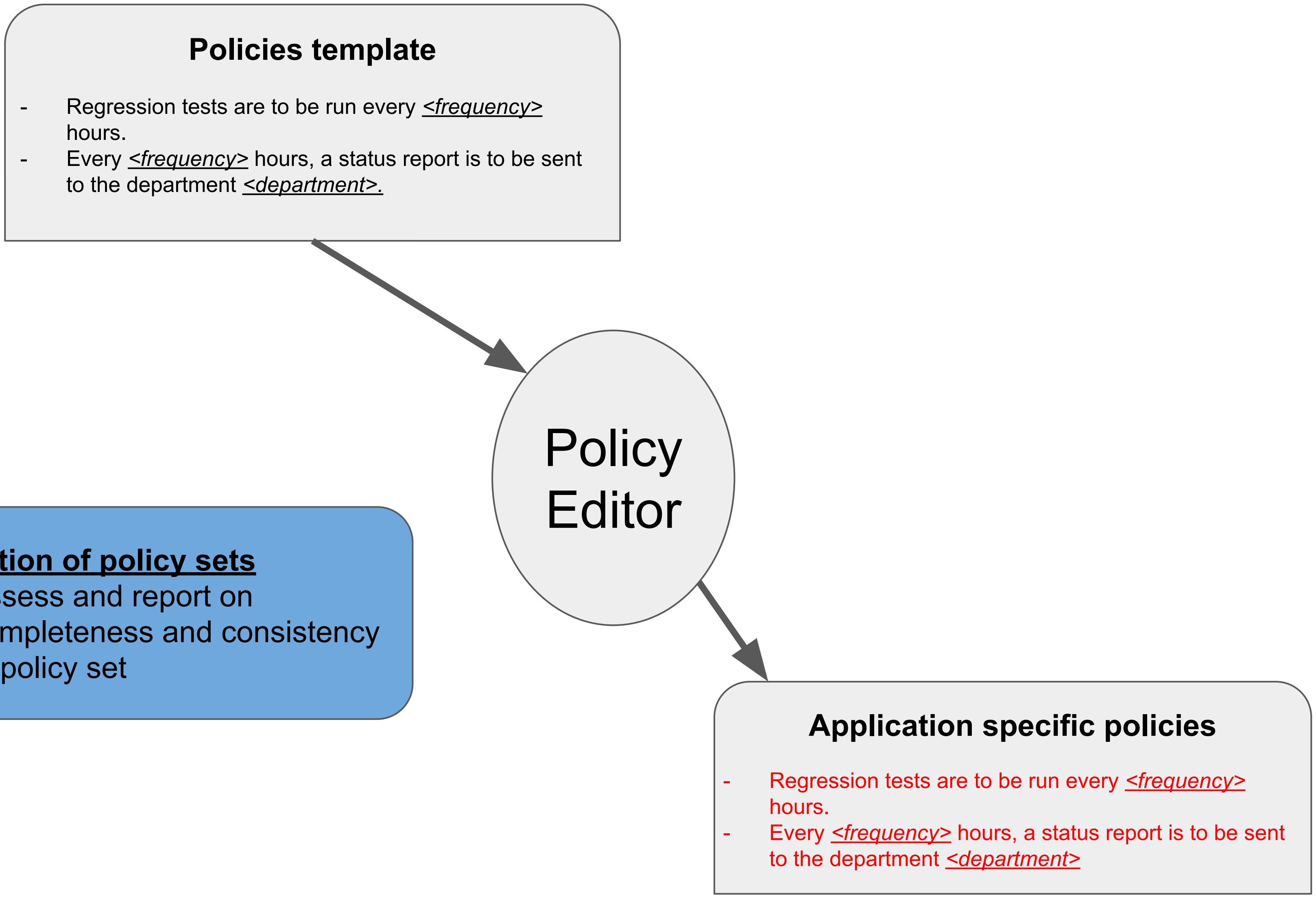
Verification of policy sets

- Assess and report on completeness and consistency of policy set

Ecosystem integration

- Filling in policies using data straight from the domain model

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*

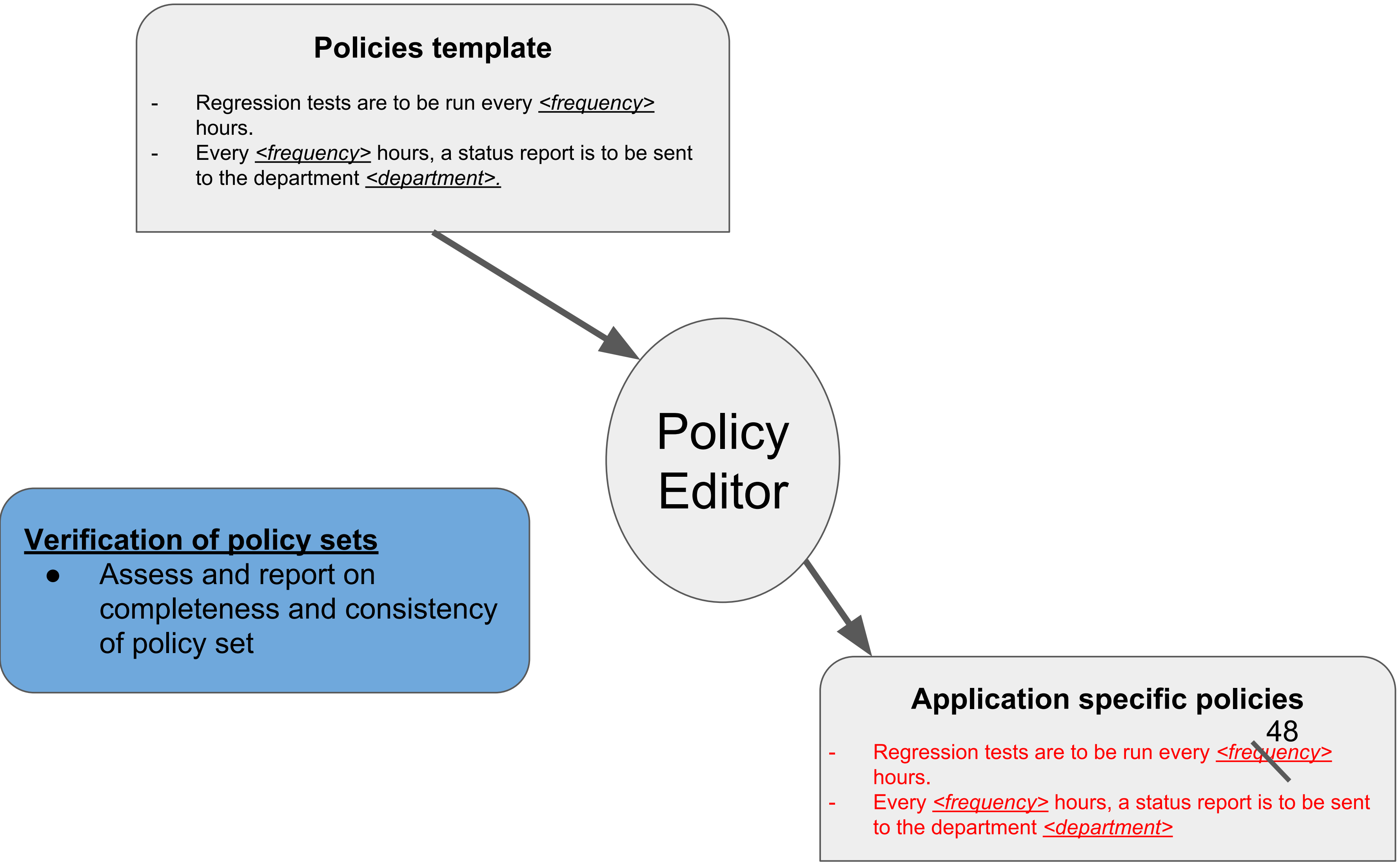
Application specific policies

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department></u>*

<u>Verification of policy sets</u>

- Assess and report on completeness and consistency of policy set

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

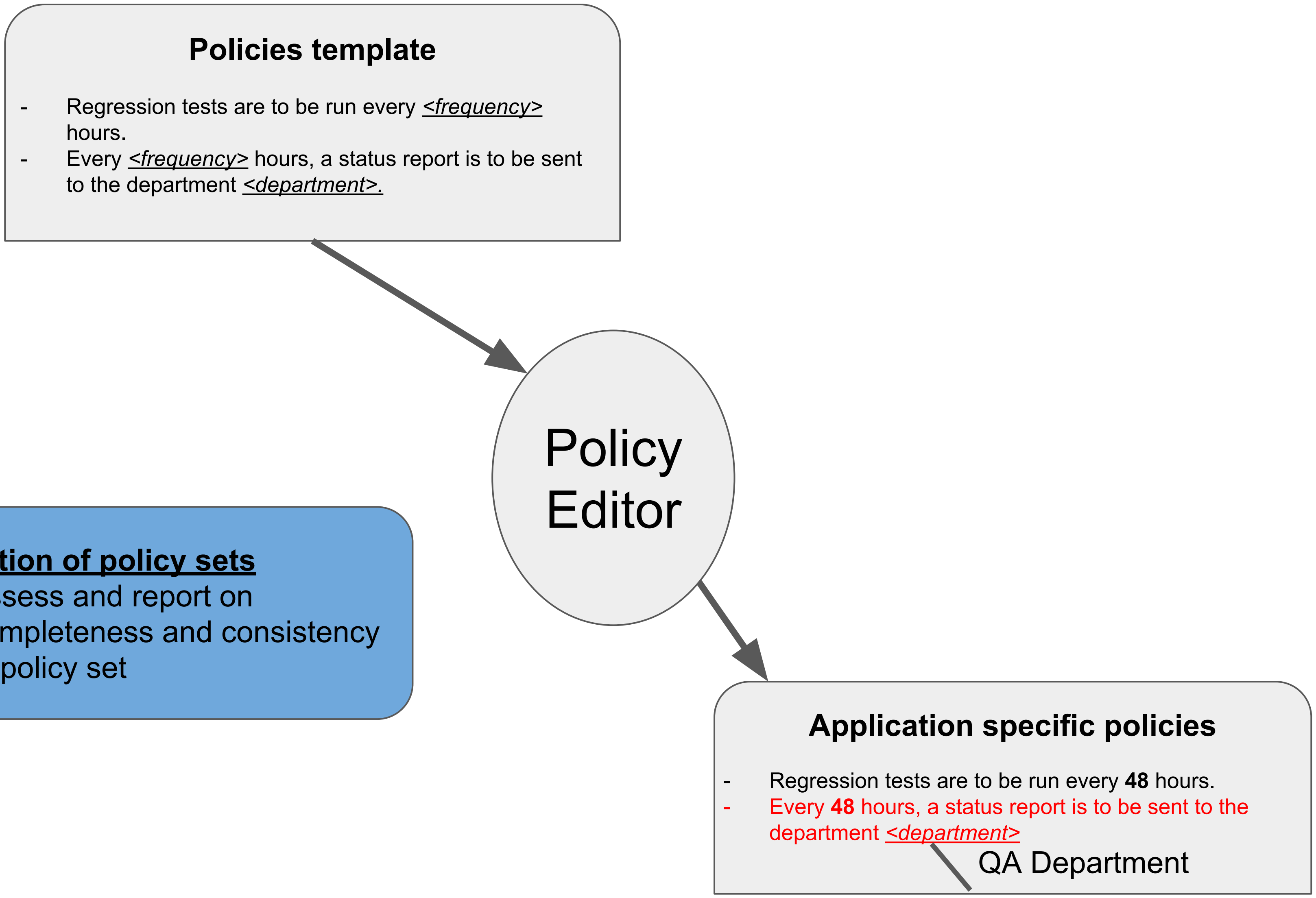
Verification of policy sets

- Assess and report on completeness and consistency of policy set

Application specific policies

- Regression tests are to be run every 48 ~~*<frequency>*~~ hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>*

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<frequency>* hours.
- Every *<frequency>* hours, a status report is to be sent to the department *<department>.*

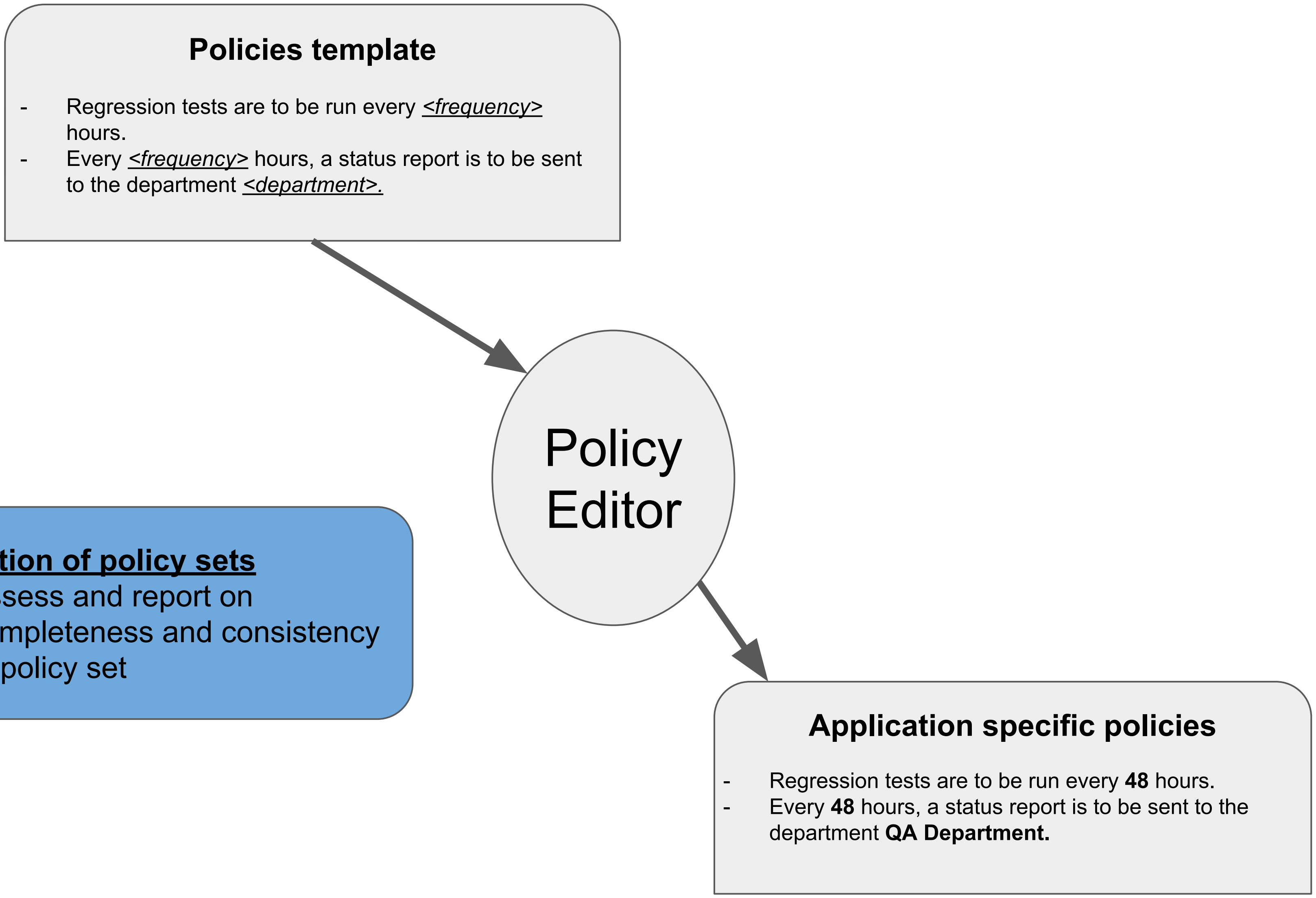
Application specific policies

- Regression tests are to be run every **48** hours.
- Every **48** hours, a status report is to be sent to the department *<department>*

QA Department

Verification of policy sets
- Assess and report on completeness and consistency of policy set

Policy Editor - Main Features



Policies template

- Regression tests are to be run every *<u><frequency></u>* hours.
- Every *<u><frequency></u>* hours, a status report is to be sent to the department *<u><department>.</u>*

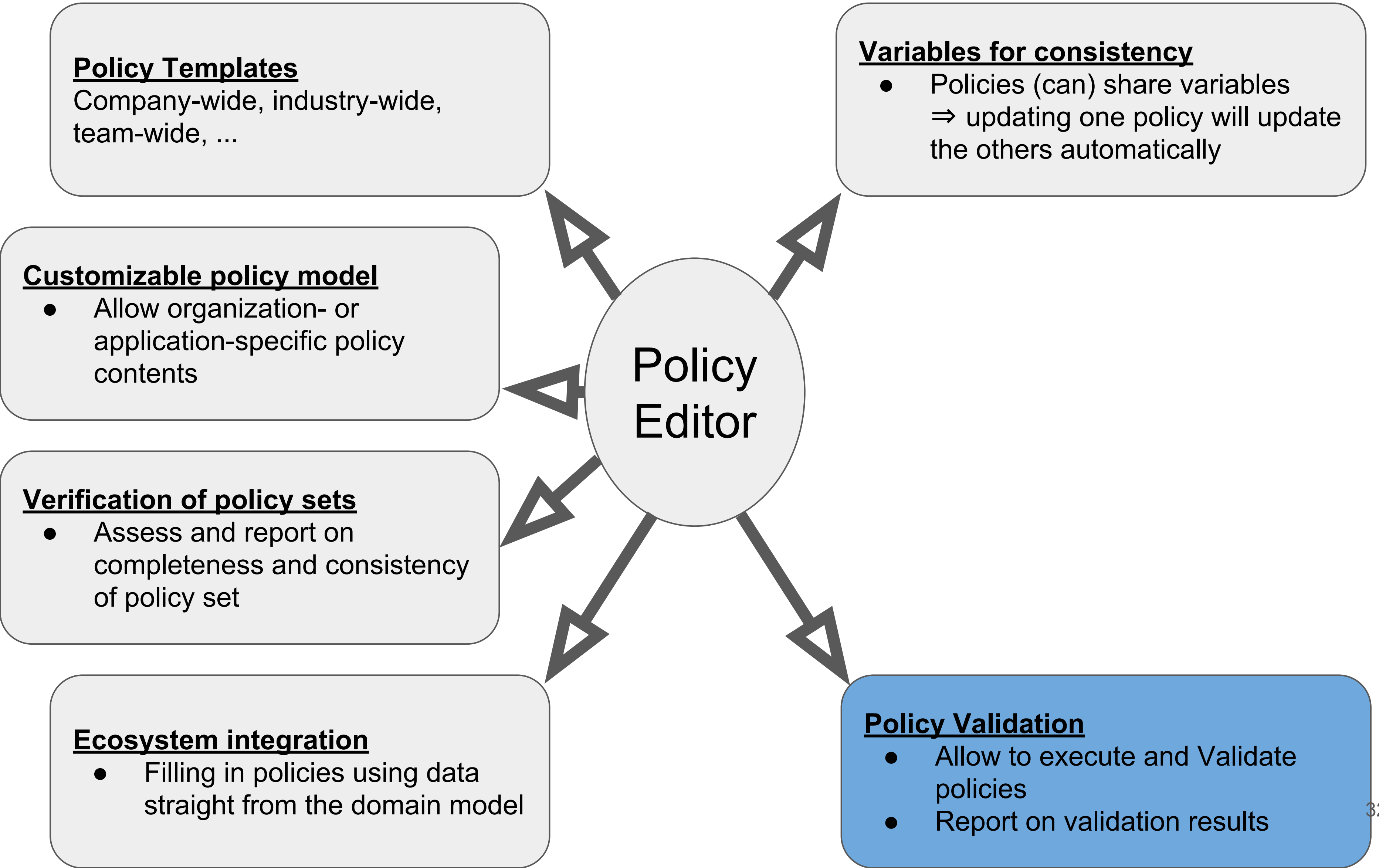
Application specific policies

- Regression tests are to be run every **48** hours.
- Every **48** hours, a status report is to be sent to the department **QA Department.**

<u>Verification of policy sets</u>

- Assess and report on completeness and consistency of policy set

Policy Editor - Main Features



Policy Templates
Company-wide, industry-wide, team-wide, ...

Variables for consistency

- Policies (can) share variables ⇒ updating one policy will update the others automatically

Customizable policy model

- Allow organization- or application-specific policy contents

Verification of policy sets

- Assess and report on completeness and consistency of policy set

Ecosystem integration

- Filling in policies using data straight from the domain model

Policy Validation

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Main Features

Application specific policies

Policy 1: Every document of type **report** must have an author

Process info:

```
For each doc in getDocumentsOfType(report):
    assert(doc.has("author"))
```

<u>Policy Validation</u>

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Main Features

Application specific policies

Policy 1: Every document of type **report** must have an author

Process info:

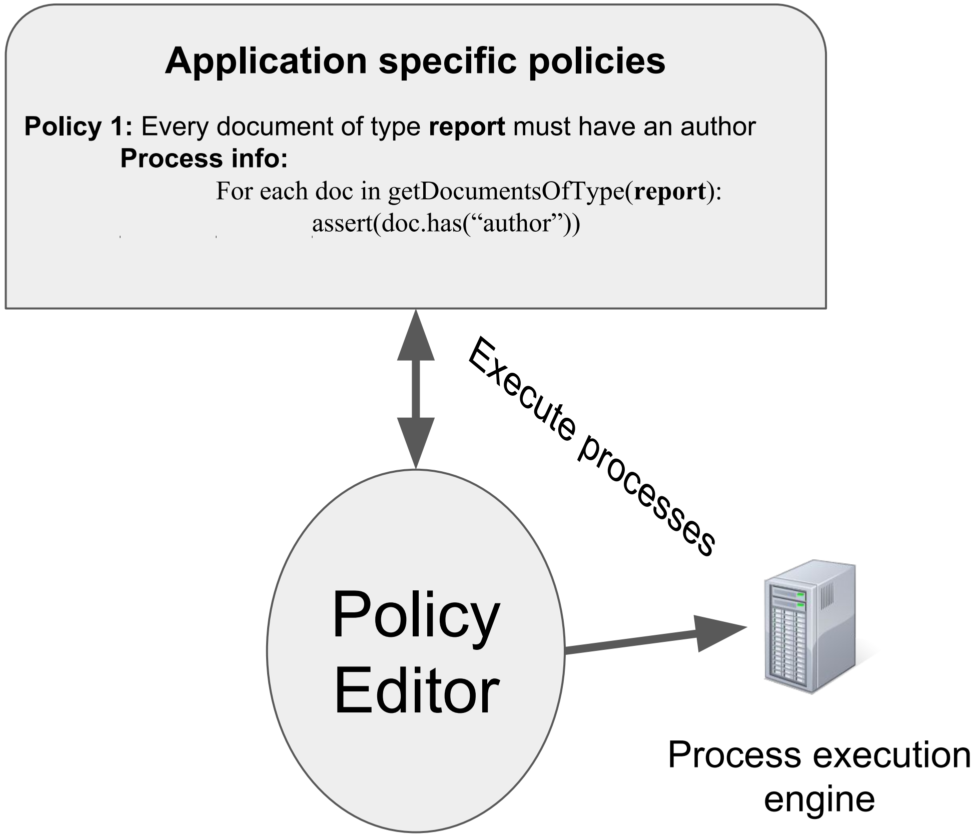
```
For each doc in getDocumentsOfType(report):
    assert(doc.has("author"))
```

Policy Editor

Policy Validation

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Main Features



Application specific policies

Policy 1: Every document of type **report** must have an author

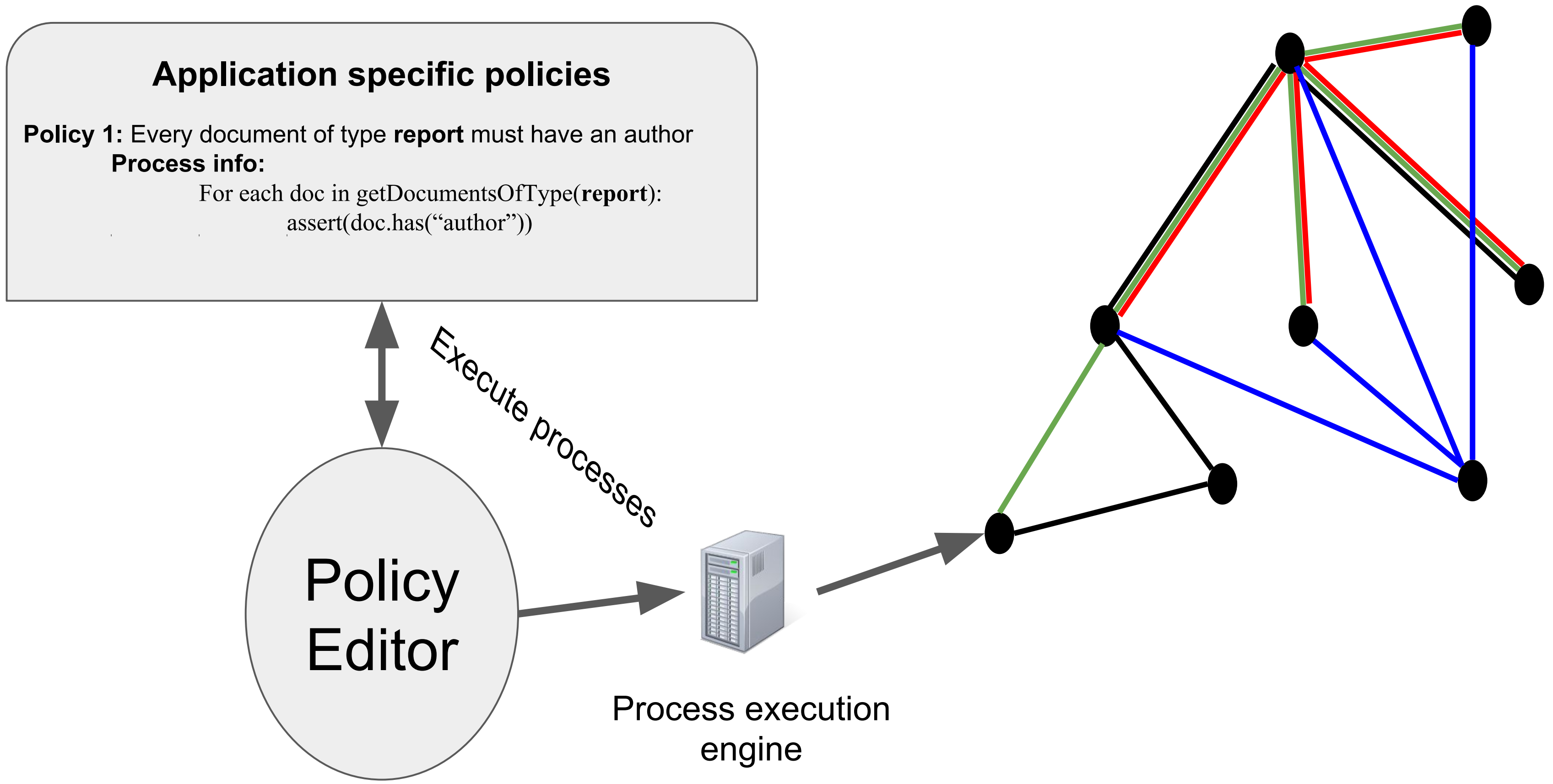
Process info:

```
For each doc in getDocumentsOfType(report):
    assert(doc.has("author"))
```

<u>Policy Validation</u>

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Main Features



Application specific policies

Policy 1: Every document of type **report** must have an author

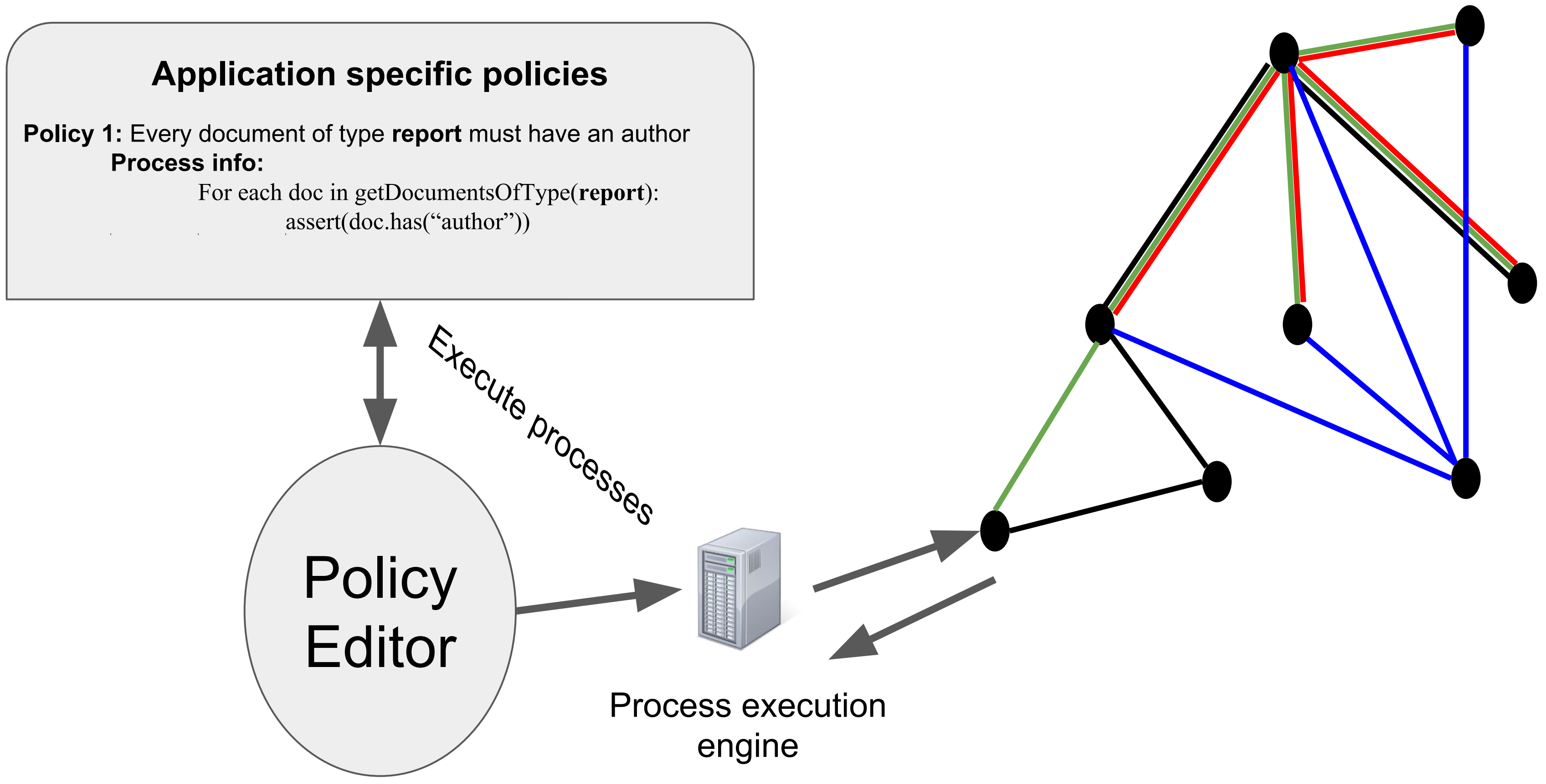
Process info:

```
For each doc in getDocumentsOfType(report):
    assert(doc.has("author"))
```

<u>Policy Validation</u>

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Main Features



Application specific policies

Policy 1: Every document of type **report** must have an author
Process info:

```
For each doc in getDocumentsOfType(report):
    assert(doc.has("author"))
```

<u>Policy Validation</u>

- Allow to execute and Validate policies
- Report on validation results

Policy Editor - Policy Model

Simple Policy Model

- Policy Text
- Author
- Version

Complex Policy Model

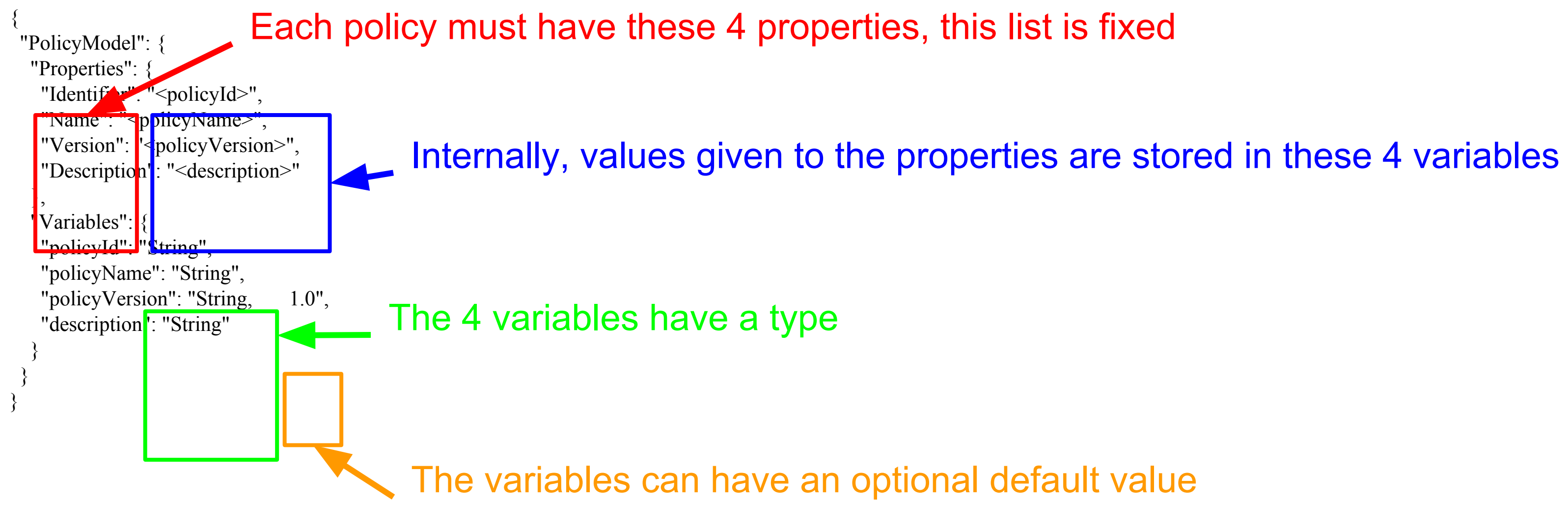
- Policy Text
- Author
- Version
- Maintainer
- Constraints
- Applicability
- Format
- Language
- Compliance
- Target users
- Replaced policies

- Policy Model determines:
  - Properties of policies
  - Possible default values of properties and values
- Policy Model currently excludes:
  - Layering structure of the policies
    ⇒ currently up to three levels of policies can be defined
    ⇒ only the lowest level policy can contain executable code

Policy Editor - Policy Model

A ‘currently active’ Policy Model is composed of up to three components:

1. **Policy Editor-specific Policy Model**
- A partial policy model that is an inherent part of the Policy Editor and that is needed for its proper functioning.
- This model is likely to be extended in the future as the Policy Editor gains additional features.
- Stored and loaded as a JSON file



```
{
 "PolicyModel": {
   "Properties": {
     "Identifier": "<policyId>",
     "Name": "<policyName>",
     "Version": "<policyVersion>",
     "Description": "<description>"
   },
   "Variables": {
     "policyId": "String",
     "policyName": "String",
     "policyVersion": "String,      1.0",
     "description": "String"
   }
 }
}
```

Each policy must have these 4 properties, this list is fixed

Internally, values given to the properties are stored in these 4 variables

The 4 variables have a type

The variables can have an optional default value

Policy Editor - Policy Model

A ‘currently active’ Policy Model is composed of up to three components:

2. **An team/organization/...-specific Policy Model**
   - A policy model that is applied on top of the Policy Editor model and can override the default values of the variables of the Policy Editor Policy Model.
   - Stored and loaded as a JSON file

Custom variable types

Custom variable type, default value

Custom variable type values

```
{
 "PolicyModel": {
  "Properties": {
   "Identifier": "<policyId>",
   "Name": "<policyName>",
   "Version": "<policyVersion>",
   "Level": "<policyLevel>",
   "Type": "<policyType>",
   "Statement Format": "<policyStatementFormat>",
   "Statement Language": "<policyStatementLanguage>",
   ...
  },
  "Variables": {
   "policyId": "String",
   "policyName": "String",
   "policyStatement": "String",
   "policyVersion": "String",
   "policyLevel": "PolicyLevel",
   "policyType": "PolicyType",
   "policyStatementFormat": "PolicyStatementFormat",
   "policyStatementLanguage": "PolicyStatementLanguage, SQL",
   ...
  }
 },
 "VariableTypes": {
  "PolicyType": ["mandatory", "legal requirement", "aspirational", "business driven"],
  "PolicyLevel": ["guidance", "procedure", "control"],
  "PolicyStatementFormat": ["formal", "non-formal"],
  "PolicyStatementLanguage": ["natural", "ReAL", "SWRL", "SQL"],
  ...
 }
}
```

Policy Editor - Policy Model

A ‘currently active’ Policy Model is composed of up to three components:

3. **Policy Template-specific modifications to the Model**
   - A Policy Template (see following slides) can also override or augment the active Policy Model.
   - In case of conflicting specifications of the Policy Model, the Policy Template has precedence over the team/organization/… specific model, which itself has precedence over the Policy Editor Policy Model.

Policy Editor - Policy Template

- JSON file
- Blueprints for templates
- Contains properties (eg. name, department, policy text, …)
    - The properties shown/editable in the Policy Editor is determined by the used Policy Model.
        - If the template contains fewer properties than present in the model, they are created
        - If the template contains properties not present in the model, they are ignored.
- Properties can contain one or more variables that can propagate to other properties and policies that contain the same variables.
    - Variables are identified using a variable name.
    - Multiple types of variables are supported:
        - *Local variables*: their value is only relevant to the policy that contains the variable
        - *Global variables*: their value is propagated to all other active policies and properties that contain that variable
        - *Hierarchical variables*: their value only propagates to upper and lower level policies, but not to siblings.

Policy Editor - Policy Template

- Example 1: empty Policy Template
  - 5 toplevel structures

```
{
 "TopPolicies": {
 },
 "SubPolicies": {
 },
 "Processes": {
 },
 "Variables": {
 },
 "VariableTypes": {
 }
}
```

Top-level policies

Mid-level policies

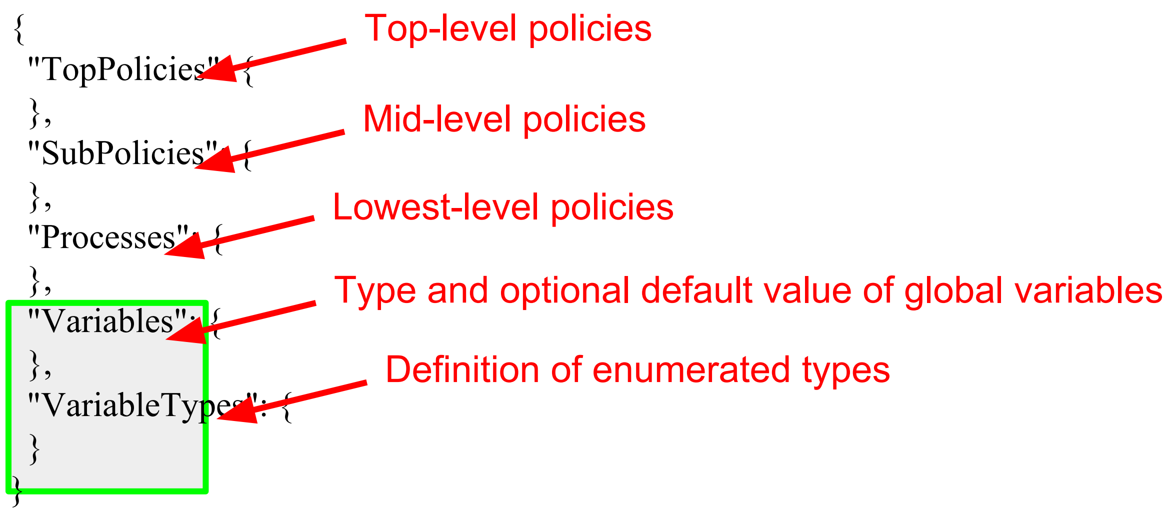
Lowest-level policies

Type and optional default value of global variables

Definition of enumerated types

Policy Editor - Policy Template

- 5 toplevel structures



```
{
  "TopPolicies": {
  },
  "SubPolicies": {
  },
  "Processes": {
  },
  "Variables": {
  },
  "VariableTypes": {
  }
}
```

Top-level policies
Mid-level policies
Lowest-level policies
Type and optional default value of global variables
Definition of enumerated types

Policy Editor - Policy Template

- Variables and Variable types

Global variables, name ends with _*_

Variable type

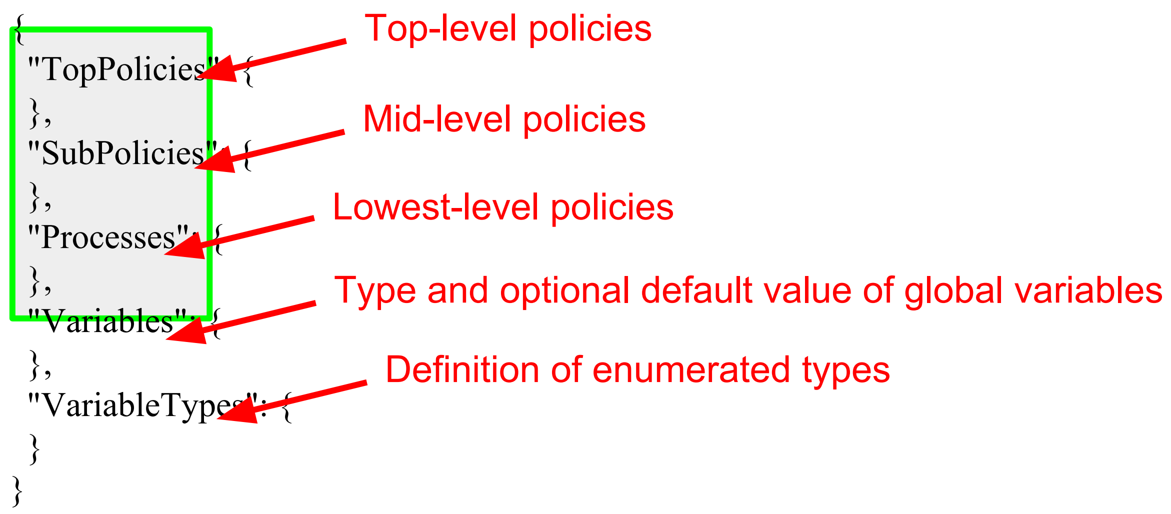
```
"Variables": {
  "repository_*_":"String",
  "scripts_location_*_": "String, /var/lib/policy-editor/scripts"
},
"VariableTypes": {
  "DataType": ["Meteosat Data", "Meteosat Data and Derived Products", "Meteosat Derived Products"],
  "DistributionForm": ["digital", "graphical", "digital and graphical"]
}
```

Optional default value

Custom data type and its enumerated values

Policy Editor - Policy Template

- 5 toplevel structures

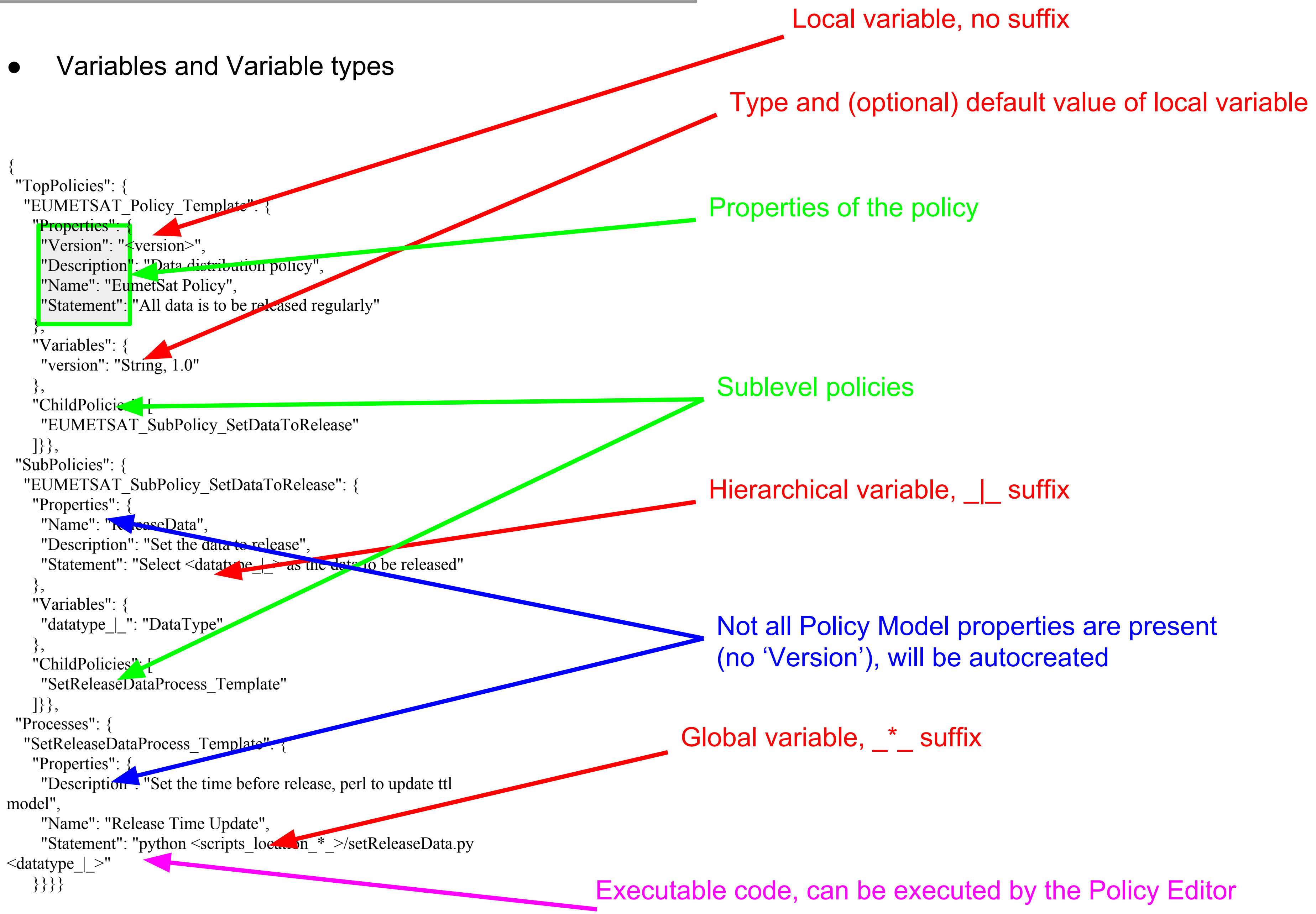


```
{
  "TopPolicies": {
  },
  "SubPolicies": {
  },
  "Processes": {
  },
  "Variables": {
  },
  "VariableTypes": {
  }
}
```

Top-level policies
Mid-level policies
Lowest-level policies
Type and optional default value of global variables
Definition of enumerated types

Policy Editor - Policy Template

- Variables and Variable types



```
{
 "TopPolicies": {
   "EUMETSAT_Policy_Template": {
     "Properties": {
      "Version": "<version>",
      "Description": "Data distribution policy",
      "Name": "EumetSat Policy",
      "Statement": "All data is to be released regularly"
     },
     "Variables": {
      "version": "String, 1.0"
     },
     "ChildPolicies": [
      "EUMETSAT_SubPolicy_SetDataToRelease"
     ]}},
 "SubPolicies": {
   "EUMETSAT_SubPolicy_SetDataToRelease": {
     "Properties": {
      "Name": "ReleaseData",
      "Description": "Set the data to release",
      "Statement": "Select <datatype_|_> as the data to be released"
     },
     "Variables": {
      "datatype_|_": "DataType"
     },
     "ChildPolicies": [
      "SetReleaseDataProcess_Template"
     ]}},
 "Processes": {
   "SetReleaseDataProcess_Template": {
     "Properties": {
      "Description": "Set the time before release, perl to update ttl
model",
      "Name": "Release Time Update",
      "Statement": "python <scripts_location_*_>/setReleaseData.py
<datatype_|_>"
     }}}}
```

Local variable, no suffix
Type and (optional) default value of local variable
Properties of the policy
Sublevel policies
Hierarchical variable, _|_ suffix
Not all Policy Model properties are present (no 'Version'), will be autocreated
Global variable, _*_ suffix
Executable code, can be executed by the Policy Editor

Policy Editor - User Interface

Please refer to the demo screencast