BugzillaMetrics Guide

Version 1.2

BugzillaMetrics Guide: Version 1.2

Abstract

This document contains information for end-users and administrators of BugzillaMetrics.

Please use our mailing list if you have questions that the guide doesn't answer. Before you post, please check the archives [http://sourceforge.net/mailarchive/forum.php?forum_name=bugzillametrics-users] . To post on the mailing list, write an email to bugzillametrics-users@lists.sourceforge.net.

The most recent version of this document can be found on the BugzillaMetrics homepage [http://www.bugzillametrics.org] .

Table of Contents

List of Tables

Chapter 1. What is BugzillaMetrics?

BugzillaMetrics is a tool for the evaluation of change request metrics. It offers the following features to the user:

- Flexible XML-based specification of a wide range of metrics (e.g. incoming rate, case life time, reopened rate, state residence time, backlog management index, etc.)
- Web-Interface for metric queries
- Graphical Wizard for metric definition without using XML
- Display of metric results as line charts or stacked area charts
- Export of the metric result to XML, Excel, HTML, and JPEG
- Traceability from metric results to individual bugs
- Easy access to the evaluation of a set of predefined common metrics
- Saving and reloading of metric queries
- Editing of XML-based metric specifications
- Access control based on existing Bugzilla user profiles or servlet-based authentication
- Optional integration with CVS/Subversion in order to evaluate metrics related to changes in the repository

BugzillaMetrics is part of the **QMetric** tool suite. The QMetric tool suite provides a generic evaluation engine for evaluating process metrics, as well as tool support for the definition of quality assessment models and their automatic evaluation.

Chapter 2. Using BugzillaMetrics

2.1. Introduction

This chapter explains the usage of BugzillaMetrics via the web interface. Internally BugzillaMetrics uses XML documents to specify how metrics are calculated (see Chapter 3, *Metric Specification*), and how the metric results are displayed in a chart (see Chapter 4, *Chart Specification*). The web interface enables the definition of metrics without using XML, as well as the evaluation of existing metrics.

The web interface is split into several tab pages:

Common Metrics	On this page you can select one of the predefined metrics for evaluation.
My Queries	This page shows a list of metric queries that had been saved by the user. These queries can be evaluated or reloaded to one of the other tab pages.
CountEvents, Count Until, Interval Length, Residence Time	These pages offer a graphical user interface that supports the detailed definition of a new metric. This enables you to define your own metrics without having to learn how to use the XML specifications.

Each page again contains a tab page *Metric* and a tab page *XML* . On the Metric tab you can select or define metrics with a graphical interface. On the XML tab you can edit the XML specifications of a metric. When selecting "Load XML" on the Metrics tab the corresponding metric specification for the selections in the Metric tab will be displayed in the XML tab. This offers the opportunity to learn how these XML specifications work. An experienced user can then edit the XML specifications if further customization of the metric definition is needed.

2.2. Basic Terms

Caution

This section explains the meaning of basic terms used within this document. Make sure to have the same understanding of these terms when you read this document.

- **Case** is used as a synonym for issues, problem reports, bugs, change requests or however you prefer to call the things administered in Bugzilla.
- **Metric** : A software metric is any type of measurement that relates to a software artifact. BugzillaMetrics enables to define a multitude of metrics related to cases administered in Bugzilla. Based on a metric definition a certain set of cases is evaluated in each time period and a number is assigned. These numbers assigned for the time periods can then be displayed as plots in a chart.

- **Query** : BugzillaMetrics allows to save metric definitions. These saved metric definitions are called queries. Queries can be evaluated directly, or reloaded to the interface and modified by the user (e.g. by changing the evaluation time period, or changing the base filter).

2.3. The Common Metrics page

The Common Metrics page lists a number of metrics with their name and description. These metrics are predefined by the BugzillaMetrics administrator. In this page you can choose a predefined metric, set additionally a number of filters (e.g. for the products of interest), select the evaluation period and granularity, and then evaluate the metric.

2.3.1. Base Filter

The selections in the upper part of the page enable to filter the cases that are considered for the evaluation of a metric. This filter is also referred to as *base filter* since it determines the base set of cases for the evaluation. All listboxes allow multiple selection. If nothing is selected in a listbox there will be no filtering for the aspect represented in the listbox. Additionally cases can be filtered for the values of flags, and it can be filtered using regular expressions.

The widgets displayed in the Base Filter can be configured by the administrator (see Configuration of the Base Filter Widget).

The filter is applied at the **end** of each time interval (e.g. each week). Only the cases that match the filter will be considered for the calculation of a metric value for this time interval.

2.3.2. Grouping Parameter

The *grouping parameter* can be used to show distinct time series for different partitions of the evaluated cases. Grouping by product for example will result in a chart with separate plots for each product. Selecting multiple grouping parameters will result in a chart with a plot for each combination of the possible values of the grouping parameters.

2.3.3. Evaluation Period and Time Granularity

The evaluation period determines the time span for which values are calculated. By selecting "always till today" and saving the query, the end date will be adjusted each time the query is reloaded.

The time granularity determines for which time intervals the data points will be calculated. Possible values are day, week, month and year. If the time granularity is week, month or year the calculation will include the whole time interval to which the start and end date belong.

Data points will always be drawn in the chart at the first day of a week, month or year. Monday is considered to be the first day of a week.

2.4. The My Queries page

This page is to manage and manipulate the saved queries. Queries can be public or private. Private Queries are visible only for the user who saved the query. Public queries are visible for all users.

Users are identified based on the login to Bugzilla itself.

2.4.1. Saving a query

Queries can be saved from each page. When saving a query the user must enter a name and a description of the query. The query name must be unique. The description of the query as well as the private/public visibility can be changed later. Public queries can only be changed by the user who initially saved them. Be aware that changing public queries might confuse other users.

2.4.2. Reloading a query

Saved queries can be evaluated directly as well as reloaded in different ways. A reloaded query can then be modified.

- The *Reload* button will reload a query to the page from which it had been saved, e.g. queries that had been saved from the Common Metrics page will be reloaded to this page. Queries that had been saved from an XML tab will be reloaded to the XML tab of the My Queries page. This is due to the fact that an XML specification can contain additional elements that can not be selected in the graphical user interface.

- The *Load XML* button will always reload the query into the XML tab of the My Queries page.

2.4.3. Static Links

BugzillaMetrics provides static links for metric results and saved queries.

To generate a static link for one or several saved queries, select these queries and click the button "Static Link". This will display a dialog with a static link. The given link will automatically trigger the metric calculation.

Metric results will also be available by a static link. To get the static link to metric results, use the button "Static Link" at the bottom of the result page. This simplifies to access metrics results again at a later point in time. Metric results are stored for a time specified by the administrator (default is 31 days).

2.5. Definition of Metrics in the Graphical User Interface

While the Common Metrics page offers easy access to a number of predefined metrics, there is also the possibility to create user-defined metrics without using XML. The metrics that can be defined with BugzillaMetrics fall in 4 different categories. Each of these categories is reflected in one of the tab pages:

Count Events	This page enables to specify metrics that are based on the occurrence of certain events (e.g. entering a new case into Bugzilla). The resulting cases can be weighted, for example by age in days or remaining work load.

Count Until	This page enables to specify metrics that are based on counting events until a specific event occurs, like calculating the number of assignee changes before resolution.
Interval Length	This page enables to specify metrics that are based on the interval length between two events, like calculating the time from creating a case to the event of transition of status to processing.
Residence Time	This page enables to specify metrics that are based on the total time a case resides in a certain status until an event occurs, like calculating the whole time a case resided in assigned status before it was resolved.

In order to define a metric you will first have to choose which of these categories is best for the definition. The categories are not fully disjoint. Due to the usage of weights in the Count Events category, is can be possible to define the same metric in two different categories.

Most of the typical metrics of interest can be defined with the graphical user interface. However, if further features are needed (like user-defined weights, or more calculation steps), this can be specified using XML.

Tip

If you have a metric in mind but you are not sure which category page to use, start off with reloading a similar metric from the My Queries page and modify it according to your needs.

Each of the category pages is split into 4 parts: Base Filter Parameters, Evaluation Time, Evaluation Specifications, and Chart Properties.

Base Filter Parameters and Evaluation Time are similar to those in the Common Metrics page (see Base Filter and Section 2.3.3, “Evaluation Period and Time Granularity”).

The Chart Properties allows to specify some characteristics of the resulting chart. The x-axis of the chart will always be the time-axis and marked according to the selected time period granularity. The label for the y-axis can be specified by the user as "Range Axis Label". Further on the result can be displayed as line chart or stacked area chart. Width and Height determine the size of the resulting chart in pixels.

The core of each metric category page is the Evaluation Specification. The next section will explain some of the common concepts that are used in all of these metric categories. The subsequent sections will explain the different category pages.

Tip

The easiest way to get used to the usage of the evaluation specifications is to reload some of the given metrics from the "My Queries" page and study their definition.

Tip

If you want to find out how a certain metric value was calculated, you can access the lists of cases that had been considered in each time interval from the HTML result.

2.5.1. Common Concepts of the Evaluation Specifications

This section will explain some concepts that are used in all of the category pages.

The evaluation of a metric in BugzillaMetrics is done in the following steps:

1. For each time interval it is determined which cases are considered for the evaluation. The set of cases depends on the base filter and event filters specified by the user.
2. A number will be assigned to a case at certain points in time (the so called *case value*). This number can be just a default number of 1, e.g. to count the number of bugs. It can be based on a weight, e.g. the age of the case. Further on it can be determined by counting certain events during the lifetime of a case, counting the days between two specified events, or counting the days a case resided in a specified state.
3. A *calculation* is applied to all case values in a time interval. The case values can for example be summed up, or their maximum or minimum value can be determined. This calculation returns a single number for each time interval.
4. Several evaluations based on step 1-3 can be defined within a metric definition . The results of these evaluations can optionally be combined using arithmetical operations.
5. The results of the evaluations will be plotted in the chart.

The following sections will explain some of the mentioned terms in more detail.

2.5.1.1. Event Filter

Events filters are use to specify the set of cases that are considered by the evaluation. The "event filters" list box allows multiple selection. If a case matches one of the selected events in the event filter listbox, it will be considered in the calculation of the metric. The following event filters are available

Case Creation	All cases that had been created within the time interval and match the BaseFilter will be considered in the evaluation.
All cases at end of time interval	All cases that exist at the end of the time interval and match the BaseFilter will be considered in the evaluation.
Added cases to basefilter	All existing cases that entered the BaseFilter within the time interval will be considered. Cases that already match the base filter at time of their creation won't be considered. Example: The basefilter filters for *component1* and an already existing case is moved to *component1* . This case will then be considered in the evaluation.
Removed cases from basefilter	All cases that left the BaseFilter within the time interval will be considered. Example: The basefilter filters for *component1* and an already existing case that was

	allocated to *component1* is now moved to *component2* . This case will then be considered in the evaluation.
Comment was added	All cases where a comment was added will be considered in the evaluation. The description entered during creation of the case will not be considered by this event filter.
Transition	This event filter can be used to filter for cases that changed their assignee or their status within a time interval. When filtering for status changes two sets of status values can be selected. The first set specifies the status before the transition ("from"), the second set specifies the status after the transition ("to"). If there is one or more "from" elements, the transition must be from one of these values to another value. If there is one or more "to" elements, the transition must be to one of these. If no "from" element is specified, the transition can be from any value, and the same with "to" elements. The creation of a case will not be considered as a status transition. Use the "case creation" event instead.
Additional filters	All cases that match the base filter and the selected additional filters will be considered in the evaluation. Selecting this option will open a dialog box that enables the selection of the additional filters.

All filters selected within the event filter listbox are OR-connected.

Caution

The event filter "additional filter" should usually be AND-connected with other event filters. Otherwise the filter will be checked for any incoming event.

2.5.1.2. Weight

While the usage of the event filters just determines a set of cases that is considered in the evaluation, weights are used to assign a number to each of these cases. The following weights are available:

Default	Assigns 1 to each case.
Age in days	Assigns the age of the case in days.
Original effort estimation accuracy	This will assign a a value based on how precise the initial effort estimation for a case is compared to the actual effort required to close it. The result will be a floating point number. It is calculated as:

$$1 - \min(1, \frac{|\mathrm{originalEstimatedEffort} - \mathrm{actualEffort}|}{\mathrm{originalEstimatedEffort}})$$

Estimated remaining Effort	Assigns the estimated remaining effort of the case.

2.5.1.3. Calculation

A number for each time interval is calculated based on the case values (i.e. the numbers assigned to the cases considered in a time interval). The following calculations are available:

Sum	Sums up the case values. Example: If the "Count Events" Evaluation and the default weight of 1 is used, this will result in the number of events considered in the time interval.
Max	Returns the maximal value of the case values. Example: If the "Count Events" Evaluation and the "Age in days" weight is used, this will result in the maximal age of the cases considered in the time interval.
Min	Returns the minimal value of the case values.
Median	Returns the median value of the case values.
Average	Returns the arithmetical mean value of the case values.
Count case values	Returns the number of case values. If the default weight of 1 is selected, the result will be equals to using the "sum" calculation.
Count cases	Returns the number of cases that contribute a least a case value. Cases which contribute several case values are counted only once.

2.5.1.4. Additional Evaluation

An additional evaluation can be added and combined with the first evaluation. An example is the "backlog management index" that is calculated as the ration between incoming and outgoing cases in a time interval. Its definition on the Count Events page consists of an evaluation that counts the incoming cases, a second evaluation counts the outgoing cases. Then the result of the first evaluation is divided by the result of the second evaluation. The following combinations are available:

Add	Adds the results of the first and second evaluation.
Subtract	Subtracts the result of the second evaluation from the result of the first evaluation.
Multiply	Multiplies the results of the first and second evaluation.
Divide	Divides the results of the first evaluation by the result of the second evaluation.
Percentage of	Calculates the percentage of the first evaluation to the second evaluation.
Don't combine	Both evaluations will be shown in the chart as separate plots.

2.5.2. Count Events

Here you can specify metrics based on selecting cases on which a certain event had occurred in a time interval.

To define a metric you have to choose an event filter and you have to choose a weight. The event filter specifies which case have to be considered in each time interval. Then a number will be assigned to each case. This number is called *case value* . The case value is determined by the weight (see Section 2.5.1.2, “Weight”). In each time interval the end result for the metric will then be determined by the given calculation, e.g. the sum of the case values or the average of all case values.

An example metric specified from the page is the number of incoming cases (incoming rate). In order to count the incoming cases you have to consider the following events:

- A new case is entered into Bugzilla. This is reflected in the "Case Creation" event.
- A case has changed some property like the product or component and is therefore within the base filter. This event is described by "Added case to base filter".
- A case is reopened. This is described by a "Transition" from the status {"Resolved", "Verified", "Closed"} to the status "Reopened".

You have to select the corresponding items in the "Event filter" list. The items are implicitly connected by OR.

Since the incoming cases should just be counted, you can use the default weight. This will assign 1 to each case value. Then the case values are summed up by selecting the calculation "sum".

Reload the metric from the "My Queries" page to see the definition in the "Count Events" page.

Tip

The metrics definitions from the "Count Events" are quite flexible due to the usage of the weights. If you have a metric in mind that is not related to measuring the lengths of some time spans in the life time of cases, you should probably start from the "Count Events" page.

2.5.3. Count Until

Here you can specify metrics based on counting how often some event occurs for a certain case *until* a another specific event takes place.

To define a metric you have to choose two event filters denoted as "Event Count" and "Until". The first one specifies the events that are to be counted in the evaluation. The second event filter specifies the event on which the counting will stop.

A case will be considered in the evaluation at the point of time when the "Until" event occurs on this case for the **first time**. Then the number of occurrences of the event to be counted will be assigned as the case value. In each time interval the end result for the metric will then be determined by the given calculation, e.g. the maximum of the case values or the average of all case values.

An example metric specified from the page is the number of assignee changes before resolution. The first event filter will say that you want to count the number of assignee changes (a transition occurs on the field assignee). The "Until" filter will say that counting ends when the status changes to "Resolved", specified as a status transition from the set {"New","Assigned"} into one of the states {"Resolved", "Verified", "Closed"}.

Reload the metric from the "My Queries" page to see the definition in the "Count Until" page.

2.5.4. Interval Length

Here you can specify metrics based on the length of a time interval between the occurrence of two specified events.

To define a metric you have to choose two event filters denoted as "From" and "To". The first one specifies the beginning of the time interval. The second event filter specifies the end of the time interval.

A case will be considered in the evaluation **each time** when the "To" event occurs on a case. Then the time in days between the occurrences of the events will be assigned as the case value. In each time interval the end result for the metric will then be determined by the given calculation, e.g. the maximum of the case values or the average of all case values.

More advanced options for the Interval Length Calculation can be configured in the XML metric specification (see Section 3.5.3, “Interval length between two events”).

An example metric that can be specified from this page is "Age at Resolution". The "From" event filter will describe the creation of a case. The "To" event filter will say that the interval ends when the status changes to "Resolved", specified as a status transition from the set {"New", "Assigned"} into one of the states {"Resolved", "Verified", "Closed"}. Then the calculation will define that we are interested in the average age of the resolved cases. Reload the metric from the "My Queries" page to see the definition in the "Interval Length" page.

2.5.5. Residence Time

This evaluation will calculate the total time in days a case stayed in a certain state before an event occurred. This total time can include more than one time span. It sums up all time spans in which the case was in the specified state during its life time until the event takes place.

To define a metric you have to define the state of interest by specifying one or multiple status values. Then you have to define the event after which a case will no longer be considered.

A case will be considered in the evaluation **each time** the specified event occurs on a case. Then the total time in days the case had resided in that state will be assigned as the case value. In each time interval the end result for the metric will then be determined by the given calculation, e.g. the maximum of the case values or the average of all case values.

Tip

If you want to know the residence time of all cases of a certain state in each time interval, choose "All cases at the end of time intervals" in the event filter.

An example metric that can be specified from this page is "Residence Time in Status New". The selection of "state" will describe which status values should be considered when summing up the residence time. For this metric only the status "New" needs to be selected. The event filter will define when a case will no longer be considered, specified as a status transition from the set {"New","Assigned"} into one of the states {"Resolved", "Verified", "Closed"}. Reload the metric from the "My Queries" page to see the definition in the "Residence Time" page.

Caution

To clarify the difference between the "Residence Time" and the "Time Interval" evaluation let's use the following examples:

- "Residence Time" in state "NEW" until the case status changes to "ASSIGNED".
- "Time Interval" between creation of case and the status change to "ASSIGNED".

The "Residence Time" evaluation will sum up several time spans in which the case had status "NEW". The "Interval Length" evaluation will only consider the first time span until the case changed to "ASSIGNED" for the first time. It will not include later time spans when the case changes back to status "NEW".

Chapter 3. Metric Specification

This chapter explains how metrics are specified using XML.

3.1. Root Element

The root element of the metric specification is a metric element. It must contain several child elements which are explained in the following sections.

```
<metric>
  <baseFilter>...</baseFilter>
  <groupingParameters>...</groupingParameters>
  <groupEvaluations>...</groupEvaluations>
  <caseValueCalculators>...</caseValueCalculators>
  <evaluationTimePeriod>...</evaluationTimePeriod>
  <timePeriodGranularity>...</timePeriodGranularity>
  <fixedFields>...</fixedFields>
</metric>
```

3.2. Base Filter

The base filter determines which cases are evaluated by the algorithm. The set of cases is defined by a state filter element inside the base filter element.

```
<baseFilter>
  One state filter XML element
</baseFilter>
```

3.3. Grouping Parameters

Grouping parameters determine how the set of case is split into different partitions that are separated in the evaluation results. One option is to do no splitting:

```
<groupingParameters>
  <none />
</groupingParameters>
```

The other option is to define one or more parameters for fields that are split. The order of the parameter definitions determines by what parameter the set of cases is split first.

```
<groupingParameters>
  <fieldGrouping>product</fieldGrouping>
  <fieldGrouping>component</fieldGrouping>
</groupingParameters>
```

3.4. Group Evaluations

Group evaluations defined how the results from the case value calculation are combined into metrics. There are two basic types of group evaluations, the details evaluation and calculations. Each group evaluation has a name that identifies the corresponding entries in the metric result. Multiple group evaluations can be defined in the groupEvaluations XML element, but they must have distinct names.

3.4.1. Details Evaluation

The details evaluation shows all case values for a certain case value calculation. In the next example, a details evaluation for the case value calculator "incoming rate" is shown. The evaluation has the name "incoming rate", too.

```
<details name="incoming rate"
         caseValueCalculator="incoming rate" />
```

3.4.2. Calculations

Calculations use mathematical operations on the results of the case value calculation to calculate a number for the metric in a flexible way.

Operations based on sets of case values allow accessing the results from the case value calculation. The following case value set based operations are available:

- **Count**. Counts the number of case values in the set, which is equal to the set size.
- **CountUnique**. Counts the number of cases corresponding to case values in the set. Cases which contribute several case values are counted only once.
- **CountBelowThreshold**. Counts the number of cases values that are below the threshold specified in the attribute "threshold".
- **CountAboveThreshold**. Counts the number of cases values that are above the threshold specified in the attribute "threshold".
- **Sum**. Sums up all case values in the set.
- **SumBelowThreshold**. Sums up the cases values that are below the threshold specified in the attribute "threshold".
- **SumAboveThreshold**. Sums up the cases values that are above the threshold specified in the attribute "threshold".
- **Maximum**. Returns the biggest case value in the set.
- **Minimum**. Returns the smallest case value in the set.
- **WinsorizedMean**. Involves the calculation of the arithmetical mean after replacing given parts of a probability distribution or sample at the high and low end with the most extreme

remaining values, typically discarding an equal amount of both. The percentage of values that are replaced at the low end and the high end has to be given in the parameters "lowEnd" and "highEnd".

These operations need to be parameterized with the identifier of a case value calculation. The following example sums the values from the case value calculator with the ID "incoming rate".

```
<sum caseValueCalculator="incoming rate" />
```

The constant operation is available for using constants in the calculation:

```
<constant>100.0</constant>
```

Binary operations can be used to combine results from other operations. The two child elements are the subcalculations that provide the results that are used as input for the binary operation. The available binary operations are:

- **Add**
- **Subtract**
- **Divide**
- **Multiply**

The following example shows the calculation of defect rate, which is the percentage of bugs in bugs and feature requests. It uses all three different kinds of operations.

```
<calculation name="defect rate">
  <divide>
    <multiply>
      <constant>100.0</constant>
      <sum caseValueCalculator="bugs" />
    </multiply>
    <sum caseValueCalculator="bugs and enhancements" />
  </divide>
</calculation>
```

3.5. Case Value Calculations

Case value calculations calculate case values on certain events. In the caseValueCalculators XML element, multiple case value calculations can be defined, but they must have distinct names.

The following case value calculations are available:

- **Count events**
- **Count number of events until another event happens**
- **Interval length between two events**
- **Time a case was in a certain state**

They are explained in the following.

3.5.1. Count Events

This calculator calculates a case value on an event specified by an event filter . The calculation is based on the weight the calculator is parameterized with. The following snippet shows the XML element for the count events calculator:

```
<countEvents id="id">
  <event>
    an event filter...
  </event>
  <weight>
    a weight...
  </weight>
</countEvents>
```

3.5.2. Count number of events until another event happens

This calculator calculates counts the number of times an event has occurred for a case until another event happened. Both events can be specified by an event filter . The following snippet shows the XML element for the calculator:

```
<countEventsUntil id="id">
  <event>
    an event filter...
  </event>
  <until>
    an event filter...
  </until>
</countEventsUntil>
```

3.5.3. Interval length between two events

This calculator calculates the length of the time interval in days between two events that happen on a case. Both events can be specified by an event filter . The following snippet shows the XML element for the calculator:

```
<intervalLength id="id">
  <from>
    an event filter...
  </from>
  <to>
    an event filter...
  </to>
  <threshold thresholdInDays="7" useThresholdWeight="true" />
  <considerToEvent>eachTime<considerToEvent>
</intervalLength>
```

The "**threshold**" tag is optional. When this tag is used, the counting the length of the time interval will be stopped if the specified "**to**" event did not happen within this period. If the attribute "useThresholdWeight" is false, the number of days since the "**to**" event will be returned, but at most the days given in the threshold. If "useThresholdWeight" is true, the value 1 will be returned for cases that exceed the threshold, otherwise 0.

Caution

Be aware that there is a difference between the "threshold" tag and the "countBelowThreshold" tag described in section Section 3.4.2, “Calculations” . Using the "threshold" tag will affect the point in time when the case value is generated, as well as the case value itself. The "countBelowThreshold" tag will only affect the aggregation of case values defined in a calculation.

The "**considerToEvent**" tag is optional. Using this tag enables to specify when case values are created. Possible values are:

- **firstTime**: A case value will be created in the evaluation at the point of time when the "To" event occurs on a case for the first time.
- **eachTime**: A case value will be created in the evaluation each time the "To" event occurs on a case. This setting is used as default if the "considerToEvent" tag is not used.
- **lastTime**: A case value will be created in the evaluation at the point of time when the "To" event occurs on a case for the last time.

3.5.4. Time a case was in a certain state

This calculator calculates the time in days a case was in a certain state before the time point of a certain event. At such an event, a case value is calculated. The event can be specified by an event filter and the state can be specified by a state filter . The following snippet shows the XML element for the calculator:

```
<stateResidenceTime id="id">
  <state>
    an state filter...
  </state>
  <event>
    an event filter...
  </event>
  <considerEvent>eachTime</considerEvent>
</stateResidenceTime>
```

The "**considerEvent**" tag is optional. Using this tag enables to specify when case values are created. Possible values are:

- **firstTime**: A case value will be created in the evaluation at the point of time when the specified event occurs on a case for the first time.
- **eachTime**: A case value will be created in the evaluation each time the specified event occurs on a case. This setting is used as default if the "considerEvent" tag is not used.

- **lastTime**: A case value will be created in the evaluation at the point of time when the specified event occurs on a case for the last time.

3.6. Evaluation Time Period

The evaluation time period element defines the time period that is evaluated by the algorithm. Currently, only a fixed time period with a start and an end date can be defined. They dates have to be specified in the format "YYYY-MM-DD". An example is given below:

```
<evaluationTimePeriod>
   <timePeriod>
     <start>2006-08-14</start>
     <end>2006-08-27</end>
   </timePeriod>
 </evaluationTimePeriod>
```

3.7. Time Period Granularity

The time period granularity defines how the evaluation time period is split into the different intervals that are evaluated in the group evaluation phase. Possible values are day, week, month, and year. An example for a month time granularity is given here:

```
 <timePeriodGranularity>
   <month />
 </timePeriodGranularity>
```

3.7.1. User-defined Time Period Granularity

Time periods can also be defined by the user to adjust metrics to release dates or business years.

```
 <timePeriodGranularity>
   <customGranularity>
      <aggregateAt>2006-01-15</aggregateAt>
      <aggregateAt>2006-08-16</aggregateAt>
      <aggregateAt>2006-12-03</aggregateAt>
      <aggregateAt>2007-10-03</aggregateAt>
   </customGranularity>
 </timePeriodGranularity>
```

3.8. Fixed Fields

In the fixed fields element, the fields for which the calculation of correct historic values should be disabled and the current values should be used instead can be specified. In the following example, the product and component fields are fixed.

```
 <fixedFields>
   <field>component</field>
   <field>product</field>
```

```
  </fixedFields>
```

3.9. Weights

Weights are used in the count events case value calculator to calculate a value for a case state. The following weights are available:

- **Default** (XML Element: <default />). Returns a fixed "1" as result for a case state.
- **Age** (XML Element: <ageInDays />). Returns the age of the case in days.
- **Days beyond deadline** (XML Element: <daysBeyondDeadline />). Returns the time in days that passed since the deadline of a case. If the deadline is not reached, the result will be negative. If a case has no deadline, the result will be 0. This weight is usually used with filtering the "beyondDeadline" field.
- **Original effort estimation accuracy** (XML Element: <originalEffortEstimationAccuracy / >). Calculates the original effort estimation by this formula: 1 - min(1, abs(originalEstimatedEffort - actualEffort) / originalEstimatedEffort)
- **Actual Effort** (XML Element: <actualEffort />). Returns the hours worked an a case.
- **Original estimated effort** (XML Element: <originalEstimatedEffort />). Returns the original estimated effort.
- **Estimated remaining effort** (XML Element: <estimatedRemainingEffort />). Returns the estimated remaining effort.
- **Current estimated effort** (XML Element: <currentEstimatedEffort />). Returns the current estimated effort (i.e. the sum of the actual effort and the estimated remaining effort).
- **Complete** (XML Element: <complete />). Returns the percentage complete of a case (i.e. the actual effort divided by the current estimated effort).
- **gain** (XML Element: <gain />). Returns the original estimated effort minus the current estimated effort.
- **CommentCount** (XML Element: <commentCount />). Returns the number of comments for a case.
- **Blocks** (XML Element: <blocks />). Returns the number of cases that are directly blocked by a case.
- **Depends on** (XML Element: <dependsOn />). Returns the number of cases to which a case directly depends on.
- **Map field values** (XML Element: <mapping />). The map field values weight maps a finite set of values from one field to numbers. It can be defined by defining the result for each value. A configuration that shows how such a mapping might look for the priority field is shown below.

  ```
  <mapping field="priority">
    <map from="P1" to="4.00" />
  ```

```
  <map from="P2" to="3.00" />
  <map from="P3" to="2.00" />
  <map from="P4" to="1.00" />
</mapping>
```

3.10. Event Filters

Event filters define which events are accepted by case value calculators. There are several simple event filters and composed event filters that combine other event filters. The following simple event filters are available:

- **Case creation** (XML Element: <create />). Filters case creation events.
- **End of time interval** (XML Element: <endOfTimeInterval />). Filters end of time interval events based on the selected time granularity (e.g. week or month). This is very useful to trigger calculations for each case and each time interval.
- **Entering of base case set** (XML Element: <enterBaseFilter />). Filters the entering of a case into the base set.
- **Leaving of base case set** (XML Element: <leaveBaseFilter />). Filters the leaving of a case from the base set.
- **Adding a comment** (XML Element: <commentAdded />). Filters the event that an additional comment was added to a case. The description entered during creation of the case will not be considered by this event.
- **State based filtering** (XML Element: <stateFilter />). This filter contains a state filter and delegates the filtering to the case filter. This looks like the following:

  ```
    <stateFilter>
      some state filter...
    </stateFilter>
  ```

  ### Caution

  The "state filter" should usually be AND-connected with other event filters. Otherwise the filter we be checked for any incoming event.

- **Field value transitions** (XML Element: <transition />). Field value transition filters react on changes of case fields. The field that changes can be specified. Any of the fields given in Table 3.1, “Available Fields” that has historic values can be used.

  If there is one or more "from" elements, the transition must be from one of these values to another value. If there is one or more "to" elements, the transition must be to one of these. If no "from" element is specified, the transition can be from any value, and the same with "to" elements. In the following example, changes of the "status" field of a case are filtered. The changes must be from any of the states in the set containing "RESOLVED","VERIFIED","SHIPPED","CLOSED" to the state "REOPENED".

```
<transition field="status">
  <from>RESOLVED</from>
  <from>VERIFIED</from>
  <from>SHIPPED</from>
  <from>CLOSED</from>
  <to>REOPENED</to>
</transition>
```

- **Field value transitions filtered with regular expressions** (XML Element: <transitionRegExp />). This filter works like the "transition" filter, but the "to" and "from" elements are evaluated as regular expressions (non-case-sensitive).

- **and/or**: The combining event filters are the "or" and the "and" event filter. They can be used to combine other event filters. The following example shows an "or" event filter combining a "create", "transition" and "enterBaseFilter" event filter.

```
<or>
  <create />
  <transition field="status">
    <from>RESOLVED</from>
    <from>VERIFIED</from>
    <from>SHIPPED</from>
    <from>CLOSED</from>
    <to>REOPENED</to>
  </transition>
  <enterBaseFilter />
</or>
```

3.11. State Filters

State filters check whether the case of a state matches what is defined in the filter or not. There are three basic types of state filter: a null state filter, field value state filters and composed state filters. The "null" state filter is the simplest type of state filter. It accepts all cases.

```
<none />
```

The "value" state filter accepts cases that have a specified value in a field. The filter in the next example accepts cases of the product with the ID "1". For entities that have an id (e.g. products or assignees) this id will be used, otherwise the value itself will be used in the filter (see fields).

```
<value field="product">1</value>
```

The "valueRegExp" state filter accepts cases where the given regular expression matches the value of a field. BugzillaMetrics uses Java's regular expression syntax and case-insensitive matching. More information about regular expressions can be found here [http://www.regular-expressions.info/reference.html]. The filter in the next example accepts cases of all products whose name begins with "Hello".

```
<valueRegExp field="product">^Hello</valueRegExp>
```

The "flagValue" state filter accepts cases where the flag given in the attribute "field" has the given flag value. Possible flag values are "+", "-", "?" and "notSet". "notSet" will match all cases where no value for the flag is entered. The filter in the next example accepts cases where the flag "review" has the value "?".

```
<flagValue field="review">?</flagValue>
```

The "not" state filter can be used to negate a "value", "valueRegExp", or "flagValue" state filter.

The "and" and "or" state filters combine state filters. They can themselves be combined by "and" and "or", too. The filter in the next example selects all bug cases from product 1.

```
<and>
  <value field="product">1</value>
  <value field="type">bug</value>
</and>
```

3.12. Fields

The following fields can be used in the specification:

Table 3.1. Available Fields

Field	**Historic values available**	**Values used in "value" state filter**
actualEffort	y	decimal
assignee	y	user ID
beyondDeadline	y	true/false
blocks	y	int
cc	y	user ID (multi-valued field)
ccCount	y	int
classification	n	classification ID
comment	y	string
comments *	y	string
commentCount	y	int
commenter	y	user ID
commenters *	y	user ID
complete	y	decimal (percentage)
component	y	component ID
currentEstimatedEffort	y	component ID
creationTimeStamp	y	yyyy-mm-dd
deadline	y	yyyy-mm-dd
deltaTimestamp	n	yyyy-mm-dd

Field	**Historic values available**	**Values used in "value" state filter**
dependsOn	n	int
dependsOnId *	n	bug id
gain	y	decimal
id	n	bug ID
keywords	y	string
operatingSystem	y	string
originalEstimatedEffort	y	decimal
priority	y	string
product	y	product ID
qaContact	y	user ID
remainingEffort	y	decimal
reporter	n	user ID
reportingPlatform	y	string
resolution	y	string
severity	y	string
status	y	string
statusWhiteboard	y	string
summary	y	string
targetMilestone	y	string
type	y	bug/enhancement
version	y	string

* These fields are multi-valued. A filter on these fields will be matched with all existing entries (e.g. the comments or commenters of a case at a considered point in time). For performance reasons, these fields can NOT be filtered with a "transition" or "transitionRegExp" filter.

Fields with no historic values in the Bugzilla database are per default fixed field s. The third column in the table determines which values are used in the specification of the filters of a metric specification.

Chapter 4. Chart Specification

This section explains hows charts can be specified using XML.

The root element of the chart specification is called "chartConfiguration". It contains several child elements that define what should be included in the chart:

```
<chartConfiguration>
   <title>The Title of the chart</title>
   <rangeMarker>...</rangeMarker>*
   <domainMarker>...</domainMarker>*
   <chart>...</chart>+
   <width>1024</width>
   <height>768</height>
</chartConfiguration>
```

The **title** element contains a string that is used as title for the chart. There can be zero or more **domainMarker** elements. They contain markers for the domain axis. There can be zero or more **rangeMarker** elements. They contain markers for the range axis. Both marker elements are explained below. There can be one or more **chart** elements. They contain the definitions for the sub charts that are included in the chart and are explained below. The **width** and **height** elements contain positive integers for the definition of the image size.

4.1. Domain Marker Elements

Domain marker elements define markers on the domain axis that can be used to mark certain dates. A vertical line that intersects the y-axis at a specified date will be drawn into the chart.

```
<domainMarker>
  <date>2006-01-05</date>
  <label>Milestone Build 3</label>
</domainMarker>
```

The **date** element of a domain marker contains the date of the marker in the format "YYYY-MM-DD". The **label** element of a domain marker contains the text that is displayed at the marker.

4.2. Range Marker Elements

Range marker elements define markers on the range axis that can be used to mark certain thresholds. A horizontal line that intersects the x-axis at a specified threshold value will be drawn into the chart.

```
<rangeMarker>
  <value>1</value>
  <label>test</label>
</rangeMarker>
```

The **value** element defines the threshold value. The **label** element of a range marker contains the text that is displayed at the marker.

4.3. Chart Elements

Chart elements contain the definition of the sub charts that are displayed.

```
<chart>
  <calculation [cumulate="true"]>...</calculation>+
  <rangeAxisLabel>Label with the range axis description
  </rangeAxisLabel>
  <type>...</type>
</chart>
```

There must be one ore more **calculation** elements. Each contains the name of a group calculation from the metric that is used to supply the data for the chart. Optionally a calculation element can have the boolean attribute **cumulate**. If cumulate is set to "true", the resulting chart will be cumulated, i.e. the values of all previous data points will be added to the value of a given data point. The default setting is cumulate="false".

The **rangeAxisLabel** element contains the text for the label that is displayed at the range axis of the sub charts. The **type** element contains the type of the chart. "line" and "stacked" are possible.

Chapter 5. Administration of BugzillaMetrics

5.1. Installation

Installing BugzillaMetrics requires the following installation steps:

1. Database Configuration (create BugzillaMetrics custom tables and grant read-only access to the Bugzilla database)

2. Deploy BugzillaMetrics into a servlet container like Tomcat. This can be done **either** from the file release (see Deployment from the File Release) or from the source code (see Deployment from the Source Code).

3. Adjust the configuration settings of BugzillaMetrics.

If you have problems with the installation send an e-mail to the mailing list mailto:bugzillametrics-users@lists.sourceforge.net and describe your problem. Please state your bugzilla version, the version of your database, and the version of your application server (e.g. tomcat 5.5).

5.1.1. Database configuration

BugzillaMetrics works on a Bugzilla database from version 2.19.3 up to a least 4.0

BugzillaMetrics uses three custom tables to store common metrics, user-defined metric queries, and metric results. The tables can be placed in the Bugzilla database or in a separate database.

1. Create a database user for BugzillaMetrics

2. The default status workflow has changed in Bugzilla 4.0. See http://bugzillaupdate.wordpress.com/2010/07/06/bugzilla-4-0-has-a-new-default-status-workflow/. You have to setup the metric definitions depending on the status workflow of your Bugzilla installation.

   - If you use the status workflow of Bugzilla before version 4.0:

     Use BugzillaMetricsFrontend/src/setup_BugzillaMetricsDB_BugzillaBeforeVersion4.sql [http://bugzillametrics.svn.sourceforge.net/viewvc/*checkout*/bugzillametrics/bugzilla_metrics_frontend/trunk/src/setup_BugzillaMetricsDB_BugzillaBeforeVersion4.sql] to create the custom tables in the preferred location.

   - If you use the status workflow introduced in Bugzilla version 4.0:

     Use BugzillaMetricsFrontend/src/setup_BugzillaMetricsDB_BugzillaSinceVersion4.sql [http://bugzillametrics.svn.sourceforge.net/viewvc/*checkout*/bugzillametrics/

bugzilla_metrics_frontend/trunk/src/ setup_BugzillaMetricsDB_BugzillaSinceVersion4.sql] to create the custom tables in the preferred location.

3. Make sure that access rights for the BugzillaMetrics database user are set accordingly. BugzillaMetrics needs only read access to the Bugzilla database and read/write access to its custom tables.

5.1.2. Deployment from the File Release

The following description shows how to deploy BugzillaMetricsFrontend from the file release into a tomcat installation. It requires to run tomcat at least with JAVA **JRE 1.5.0_07** .

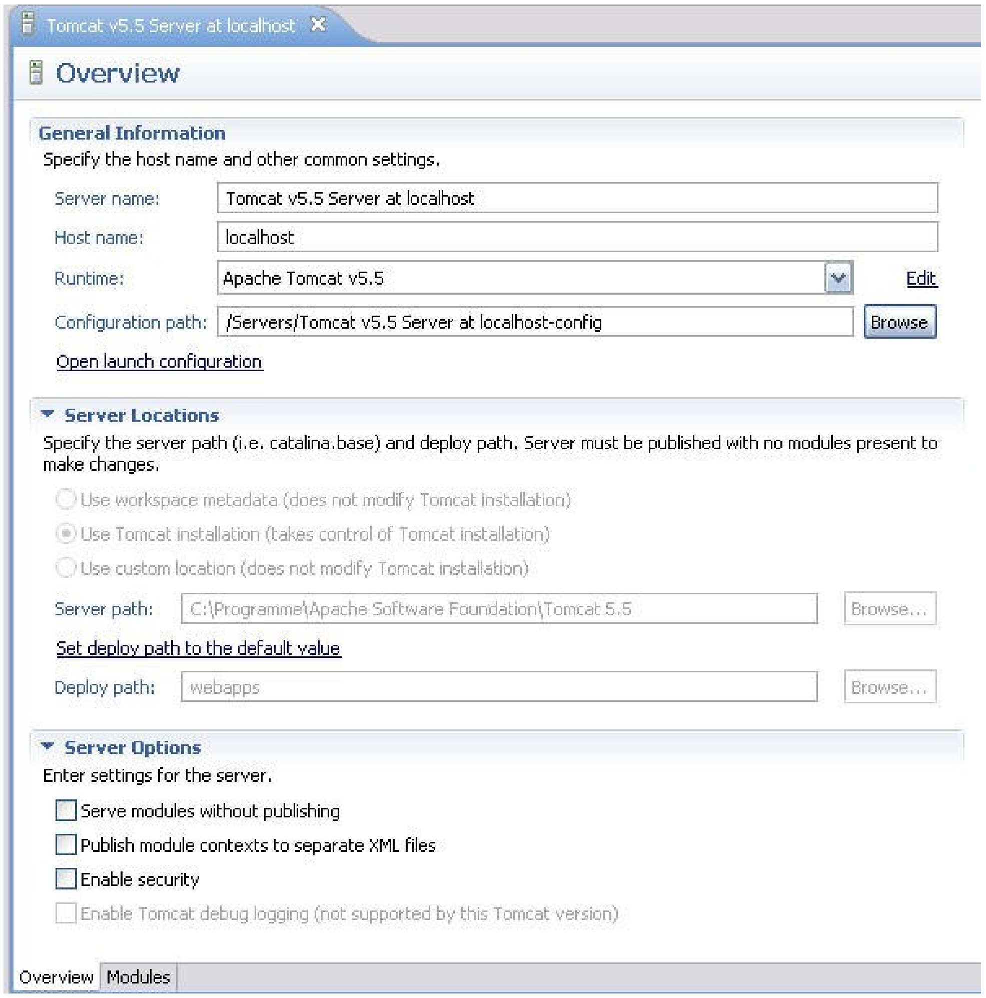
1. Download the file release [http://sourceforge.net/project/showfiles.php?group_id=197170] .
2. Unzip the file into <tomcathome>/webapps/
3. Adjust the configuration settings . The configuration files are placed in <tomcathome>/ webapps/BugzillaMetricsFrontend/WEB-INF/
4. Restart tomcat

5.1.3. Deployment from the Source Code

The following description shows how to deploy BugzillaMetricsFrontend from the source code into a tomcat installation.

It requires at least Eclipse 3.5 with WebTools Platform 2.0 and the Google plugin installed. You need to separately install the Google Web Toolkit [http://code.google.com/webtoolkit/ versions.html]. Please use GWT version 2.2 or later.

1. Get the BugzillaMetrics, BugzillaMetricsFrontend, and BugzillaMetricsDocumentation project from http://bugzillametrics.svn.sourceforge.net/svnroot/bugzillametrics [http:// bugzillametrics.svn.sourceforge.net/svnroot/bugzillametrics]
2. Set the variable "BugzillaMetricsLib" to the directory BugzillaMetrics/modules/core/lib (see Project -> Properties -> Java Build Path -> Libraries -> Add Variable).
3. Adjust the configuration settings . The configuration files are in the BugzillaMetricsFrontend project in the folder war/WEB-INF/
4. Open the package explorer in Eclipse (Window -> Show View -> Java -> Package Explorer). Select the project BugzillaMetricsFrontend, right-click to open the context menu, then click "Google -> GWT Compile". This will generate the client side javascript code. Press F5 to refresh the package explorer.
5. Open the server view in Eclipse. (Window -> Show View -> Other -> Servers -> Server)
6. Right click in the server view to open the context menu. Choose New->Server and define your server installation on localhost. The following dialog shows an example configuration.



General Information

Specify the host name and other common settings.

Server name: Tomcat v5.5 Server at localhost

Host name: localhost

Runtime: Apache Tomcat v5.5

Configuration path: /Servers/Tomcat v5.5 Server at localhost-config

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

- ○ Use workspace metadata (does not modify Tomcat installation)
- ● Use Tomcat installation (takes control of Tomcat installation)
- ○ Use custom location (does not modify Tomcat installation)

Server path: C:\Programme\Apache Software Foundation\Tomcat 5.5

Set deploy path to the default value

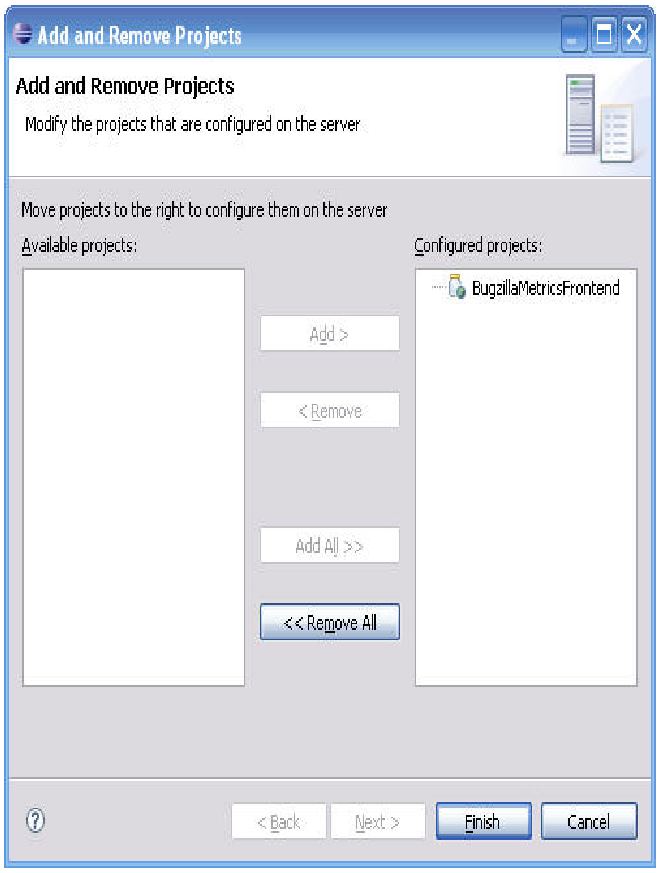
Deploy path: webapps

Server Options

Enter settings for the server.

- ☐ Serve modules without publishing
- ☐ Publish module contexts to separate XML files
- ☐ Enable security
- ☐ Enable Tomcat debug logging (not supported by this Tomcat version)

7. Choose "Add and remove projects..." in the context menu of the server view and add the BugzillaMetricsFrontend web module.



8. Choose "Publish" in the context menu of the server view.

Tip

You can also run BugzillaMetrics directly from Eclipse. Open the context menu of the package explorer and select "Run As -> Web Application". This will run the application in the GWT Hosted Mode.

5.2. Configuration Settings of BugzillaMetrics

The configuration settings of BugzillaMetrics are placed in the file **settings.xml**. It is in the WEB-INF directory. The XML tags of this file are explained in the following. Typically you only have to adjust the settings given in the tags "bugzillaDB", "metricsDB", "bugzilla" and "frontend".

- **bugzillaDB**: Connection settings to the Bugzilla database are given in the attributes database, user, password and host of this tag.
- **metricsDB**: Connection settings to the database containing the BugzillaMetrics custom tables. In case that the custom tables are placed in the Bugzilla database, the settings are the same like in the tag "bugzillaDB".
- **bugzilla**: The attributes in this tag characterize the underlying Bugzilla installation.

- bugzillaVersion: Version number of your Bugzilla installation. This enables BugzillaMetrics to adapt itself to the specific database scheme. State a version number **a.b.c** as **a*10000+b*100+c** e.g. Version 2.19.3 should be specified as 21903
- useClassifications: This setting indicates whether the classifications are used. (see classifications [http://www.bugzilla.org/docs/3.0/html/classifications.html]).

- **frontend**: The attributes in this tag define the behaviour of the graphical user interface.
  - useLoginCheck: This setting indicates whether a login check should be performed before entering BugzillaMetrics. This will check if the user is logged in to Bugzilla based on Bugzilla's login cookie (see notes on authentication [http://www.ravenbrook.com/project/p4dti/tool/cgi/bugzilla-schema/index.cgi?action=single&version=3.0&view=View+schema#notes-authentication]). Using the login check requires that BugzillaMetrics runs in the same domain as Bugzilla (see the cookie specification [ftp://ftp.isi.edu/in-notes/rfc2965.txt]).
  - bugtrackerURL: The base URL of your Bugzilla installation. If the login check fails, the user will be pointed to this URL to login into Bugzilla. Additionally this URL will be used to links to bug lists from the metric results.
  - expirationTime; This setting defined how many days metric results are kept in the database and can be accessed by a static link.
  - dataProvider: This setting only needs to be changed if BugzillaMetrics is used on a different bugtracking system than bugzilla. The setting denotes the fully qualified class name of a "DataProvider" class that provides methods to retrieve base data like the list of available products and versions from the bugtracking database. The default setting is "org.qmetric.frontend.server.dataProvider.bugzilla.DataProviderForBugzilla"
  - useBugzillaLogin: This setting indicates whether the login check should happen against Bugzilla. The default value is true. If set to false, BugzillaMetrics will use authentication of the servlet container, via the HttpServletRequest.getRemoteUser() method. This is useful when an external authentication mechanism is needed. It is preferred to use Bugzilla because it requires a much simpler setup. Refer to the documentation of your servlet container for details on setting up servlet authentication.

    If using servlet authentication, BugzillaMetrics will associate queries with the return value of getRemoteUser(). In case you installed BugzillaMetrics in version 0.9.5 or earlier then you have to update the BugzillaMetrics database. This can be done with a sql command:

    ```
    use database bugzillametrics;
           alter table queries change user user varchar(255);
    ```

    As a simple example using Tomcat and an LDAP server, this is added to the <web-app> section of WEB-INF/settings.xml:

    ```

    <security-constraint>
      <web-resource-collection>
    ```

```
    <web-resource-name>Entire Application</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>

  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>

<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Bugzilla Metrics Frontend</realm-name>
</login-config>

<security-role>
  <role-name>employee</role-name>
</security-role>

<security-role>
  <role-name>contractor</role-name>
</security-role>
```

And then tomcat's conf/server.xml file is modified to use ldap:

```

<Engine name="Catalina" defaultHost="localhost">
  <Realm className="org.apache.catalina.realm.JNDIRealm"
         debug="99"
         connectionURL="ldap://yourldapserver.com:389"
         userPattern="userid={0},ou=People,
                      ou=Intranet,dc=domain,dc=com"
         userRoleName="userGroup" />
  <Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true"
        xmlValidation="false" xmlNamespaceAware="false" />
</Engine>
```

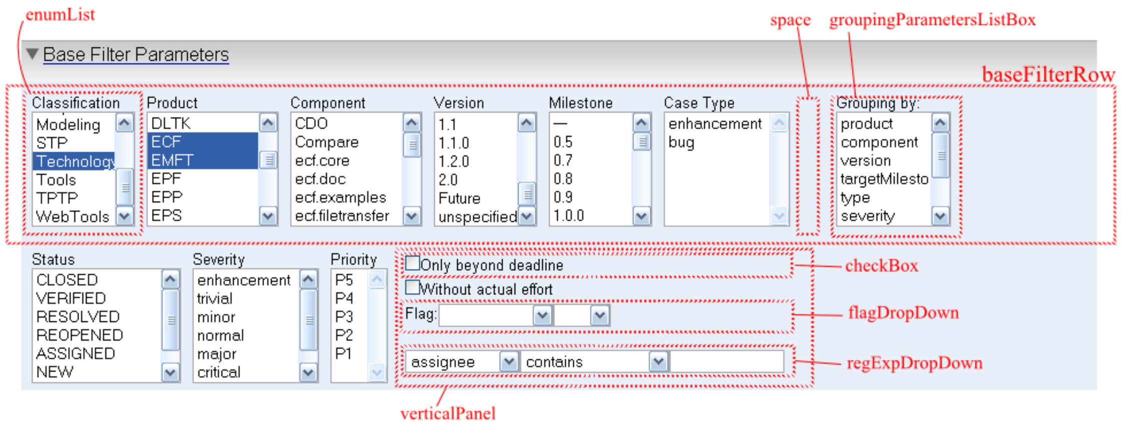
- **calculationCore**: This setting only needs to be changed if BugzillaMetrics should be used with evaluations on SCM systems or it is used on a different bugtracking system than bugzilla. This tag does contain the attribute "coreConfiguration". It denotes the fully qualified class name of a "CoreConfiguration" class that is used to configure the metric calculation component of BugzillaMetrics. The default setting is "org.qmetric.bugzilla.CoreConfigurationBugzilla" in order to run BugzillaMetrics on a Bugzilla database.

- **frontendPresentationSettings**: Some options available in the graphical user interface can be configured here, e.g. which listboxes are displayed in the "base filter" section of the frontend. The possible settings are described in detail in the next section.

The file **logging.properties** is used to configure which log messages are generated by BugzillaMetrics. The logging mechanism of BugzillaMetrics is based on Log4j (see introduction to log4j [http://logging.apache.org/log4j/1.2/manual.html]). The setting log4j.appender.A1.file=bugzilla_metrics.log can be used to specify the folder where the log file should be placed.

5.2.1. Frontend Presentation Settings

The Base Filter Widget displayed on the frontend can be configured by the administrator. The configuration is given in the tag **baseFilter** contained in the file settings.xml. The following picture shows the base filter widget with markers for the different types of elements that can appear in the widget.



The corresponding parts of the settings file are given in the following:

```
<baseFilterConfiguration>
 <frontendPresentationSettings>
    <baseFilter>
  <enumList title="Classification"
            field="classification"
            width="90px" />
  ...
  <space width="20px" />
  <groupingParametersListBox title="Grouping by:"
                             width="100px">
   <groupingParameter>product</groupingParameter>
   ...
   <groupingParameter>assignee</groupingParameter>
  </groupingParametersListBox>
 </baseFilterRow>
 <baseFilterRow>
  <enumList title="Status" field="status" width="130px" />
  ...
  <verticalPanel width="160px">
   <checkBox title="Only beyond deadline"
             field="beyondDeadline" />
   ...
   <flagDropDown title="Flag:" width="100px" />
   <regExpDropDown title="" width="100px">
    <fields width="100px">
     <field>assignee</field>
     ...
           <field>targetMilestone</field>
    </fields>
   </regExpDropDown>
  </verticalPanel>
 </baseFilterRow>
  </baseFilter>
  ...
</frontEndPresentationSettings>
```

The tags of this file are explained in the following:

baseFilterRow	The base filter widget is composed of these rows. It contains at least 2 rows.
enumList	This list enables a multiple selection in a list of enumerated values. The field name is given in the attribute "field". Each field may only appear once. The following fields can be used: classification, component, operatingSystem, priority, product, reportingPlatform, resolution, severity, status, targetMilestone, type, version.
groupingParametersListBox	This list contains the available grouping parameters.
verticalPanel	A verticalPanel is used to aggregate elements like "checkBox", "flagDropDown" and "regExpDropDown" and displays them in vertical order.
space	This element can be used to insert empty space between widgets. If this tag appears below a "baseFilterRow" use the attribute "width" to define the width of the widget. If it appears below a "verticalPanel" use the attribute "height" to define the height of the empty space.
checkBox	This element can be used to define a filter for fields with boolean values. Additionally the attribute "checkedValue" can be used to define a value that is used instead of "true" if the checkbox is selected.
flagDropDown	This element enables to define filters for flags. The attribute "width" defines the width of the first drop down list with the available flag types.
regExpDropDown	This element enables to define filters based on regular expressions. The attribute "width" defines the width of the textbox for the expression. The attribute "width" of the tag "fields" defines the width of the field drop down list.

The available fields are also defined here. Any field with id or string values can be used here.

Furtheron the events displayed in the Count Events, Count Until, Interval Length, and Residence Time category, and weights displayed on the Count Events page can be configured. These settings are typically needed if BugzillaMetrics is used with the software configuration management extension, and related events and weight should also be visible on the frontend.

5.2.2. Common Metrics

The metrics displayed on the "Common Metrics" tab page of the BugzillaMetrics frontend can be edited by changing the entries in the *metrics* database table. The typical way to do this would be to save your query from the frontend, then use an SQL statement like the following to copy this query into the *metrics* table.

```
INSERT INTO metrics SELECT name, description, query AS metric,
chart from queries where id=1234
```

5.2.3. Evaluation of custom fields

BugzillaMetrics supports the evaluation of custom fields defined in your Bugzilla installation. The following types of custom fields can be evaluated: FreeText, SingleSelect, MultiSelect, LargeTextbox, and DateTime.

To enable the evaluation of a custom field, an entry must be added to the file **metric.xsd**. The file is contained in \BugzillaMetricsFrontend\WEB-INF\lib\BugzillaMetrics.jar in the folder \org \bugzillametrics\core. (The jar-file can be opened as zip to edit the file.) The application does not need to be recompiled.

For a custom field called "cf_myCustomField" the following line must be added for the simpleType "field":

```
<xs:enumeration value="cf_myCustomField"/>
```

Caution

These restrictions apply due to the way Bugzilla stores historic values in the database: If a legal values of a field of type single-select or multi-select had been renamed by the administrator, the historic values of this field can not correctly be retrieved. If a field of type multi-select has a legal value that contains the substring ", " historic values can not correctly be retrieved.

5.2.3.1. Display filter for custom fields in the web front end

Filters for custom fields can also be displayed in the Base Filter Widget of the web front end. Fields of the type FreeText, LargeTextbox, and DateTime can be displayed in an element of type regExpDropDown. Just add the field name to the list of available fields. Fields of type Single-Select and Multi-Select can be displayed in an element of type enumList (see Configuration of the Base Filter Widget).

5.3. Environment configuration

If you have a rather large Bugzilla database (>20000 bug entries), consider to adjust the following configuration settings.

5.3.1. MySQL

Make sure that the max_allowed_packet parameter of MySQL is at least 16MB. This setting can be changed in the configuration file /etc/my.cnf (See http://dev.mysql.com/doc/refman/6.0/en/program-variables.html).

5.3.2. Java Heap Space

Increase the java heap space of the servlet container. For example add the following option to your Tomcat startup script.

```
CATALINA_OPTS=="-Xmx2048m"
```

5.3.3. Tomcat Session Timeout

On a large Bugzilla database, the session timeout may be too low for some metric calculations. To increase the timeout value in Tomcat, please locate the following section in [Tomcat_home]/conf/web.xml:

```
<session-config>
     <session-timeout>30</session-timeout>
</session-config>
```

The timeout value is specified in minutes. Restart Tomcat after modifying the file, so the changes can take effect.

5.3.4. Configuring ProxyPass on an Apache Web Server

The **ProxyPass** directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server.

If BugzillaMetrics is accessed using the ProxyPass directive, the timeout for the proxy request must be redefined. Otherwise a metric calculation may be timed out before the result is available, since some BugzillaMetrics calculation can take several minutes dependent on the size of the Bugzilla database. An example configuration is given in the following:

```
RewriteEngine on
RewriteRule ^/BugzillaMetrics$ /BugzillaMetrics/ [R]

#Timeout for proxy requests in seconds.
ProxyTimeout 3600
Timeout 3600
```

```
ProxyPass /BugzillaMetrics/
          http://localhost:8080/BugzillaMetricsFrontend/
          keepalive=on
ProxyPassReverse /BugzillaMetrics/
                 http://localhost:8080/BugzillaMetricsFrontend/

# Adjusts the domain string / path of the session cookie
ProxyPassReverseCookieDomain
                 http://localhost:8080/BugzillaMetricsFrontend/
                 public.example.com
ProxyPassReverseCookiePath / /BugzillaMetrics/
```

5.4. Miscellaneous

5.4.1. Using BugzillaMetrics with a database other than MySQL

BugzillaMetrics can be configured to run on a database other than MySQL. You have to find a JDBC driver for your database. The JDBC driver for PostgreSQL can for example be downloaded from here [http://jdbc.postgresql.org/download.html]. Copy the jar file of the JDBC driver into the directory /WEBINF/lib. Then add the attributes "jdbcdriver" (the class name of the jdbc driver class) and "subprotocol" (e.g. postgres or oracle) to the connection settings of the database given in the file settings.xml. Example:

```
<bugzillaDB
  database="bugzilla"
  user="bugmetrics"
  password="password"
  host="localhost"
  jdbcdriver="org.postgresql.Driver"
  subprotocol="postgres"
 />
```

5.4.2. ValueNotResolvedException

This exception can be thrown during a metric evaluation and will be reported in the bugzilla_metrics.log file. It is caused by inconsistencies of the Bugzilla database. Bugzilla holds the change history in a table "bugs_activity". This table does not use IDs to refer to entities like products, assignees, etc. Instead the names of these entities are used. So the entries in the "bugs_activity" table must be resolved to the ids. Renaming an entity e.g. a product cause an inconsistency, since historic changes are no more associated with this entity.

Chapter 6. Integration with CVS / Subversion

This chapter describes the integration of data from software configuration management (**SCM**) systems into BugzillaMetrics. It can be used in a software development project that uses Bugzilla and a SCM system like CVS or Subversion with bug references in commit messages. These references must be integrated into Bugzilla with the external tool SCMBug [http://www.mkgnu.net/?q=scmbug] Under these circumstances the user of BugzillaMetrics can specify metrics that combine information from Bugzilla and from SCM systems. For these purposes the SCM integration provides additional BaseFilter, Events and Weights (see Section 6.2, “Metric specification involving the SCM - Integration”).

Before SCM systems can be analyzed, the data must first be imported into a separate database that enables the metric analyzes in an efficient way. This is described in Section 6.1.4, “Import SCM data” . The required installation steps are described in the next section.

6.1. Installation

Installing SCM integration requires the following installation steps:

1. Create a new database for imported data from SCM systems and grant write access to it (see Section 6.1.1, “SCM Database configuration”).
2. Install additional tools that are needed by the import module (see Section 6.1.2, “Needed additional tools”).
3. Adjust the configuration settings for the import of CVS / Subversion data (see Section 6.1.3, “Configuration Settings”).

6.1.1. SCM Database configuration

The SCM integration works on a separate database that can be created in the following three steps:

1. Create a database user for the SCM integration or use the same user who was generated during the installation of BugzillaMetrics.
2. Use scm_sql_configuration.sql [http://bugzillametrics.svn.sourceforge.net/viewvc/*checkout*/bugzillametrics/bugzilla_metrics/trunk/modules/scm/src/org/bugzillametrics/scm/sql/scm_sql_configuration.sql] to create the scm database in the preferred location.
3. Make sure that access rights for the scm database user are set accordingly. The SCM integration needs read and write access to the scm database.

6.1.2. Needed additional tools

Make sure that you have installed the following tools:

- The Unix command line tool **diff**: Unix operating systems should have installed this tool by default. If this is not the case for your Unix system then install GNU diffutils. Users of Windows ™ - operating systems require a Windows ™ - porting of diff. This is, for example, available in the SourceForge-Project UnxUtils and can be downloaded at http://unxutils.sourceforge.net.
- A script extracted from Scmbug which translates user names between Bugzilla and SCM systems. Download it from http://bugzillametrics.svn.sourceforge.net/viewvc/bugzillametrics/scmbug_user_name_translation/trunk/UserNameTranslation/UserNameTranslation.pm

6.1.3. Configuration Settings

A file named "looging.properties" is used to configure logging during the import process. Download a template of this file from here [http://bugzillametrics.svn.sourceforge.net/viewvc/bugzillametrics/bugzilla_metrics/trunk/modules/scm/src/org/bugzillametrics/scm/dataImport/logging.properies]. Usually you only have to adjust the path to the logging file (log4j.appender.A1.file).

A file named "settings.xml" is used to configure the import from CVS or Subversion". Download a template of this file from here [http://bugzillametrics.svn.sourceforge.net/viewvc/bugzillametrics/bugzilla_metrics/trunk/modules/scm/src/org/bugzillametrics/scm/dataImport/settings.xml]. Configure the following items in your settings.xml file:

- **scmDB**: Connection settings to the scm database are given in the attributes database, user, password and host of this tag.
- **scm**:
  - **scmSourceLocation**:
    - scmRootPath: Location of the folder that contains the SCM repository.
    - cvsAccessType: Access type of CVS system (local, pserver)
    - projectName: Name of the scm project that should be imported.
  - **scmImportWorkingDirectory**:
    - absoluteWorkingDirecoryPath: Path to a local folder that should contain the working files of the import module.
    - relativeWorkingDirectoryTmpPath: Relative path in your working directory for temporary files.
    - relativeWorkingDirectoryLogPath: Relative path in your working directory for log files.
    - relativeWorkingDirectoryCheckoutPath: Relative path in your working directory for the local working copy of the import module.
  - **scmImportEventCheckHeuristics**:

  - relevantFiletypes: Filetypes that should be analyzed by import module at codeline level. It should be specified as list: java,c,html
  - maxTimeIntervalInMinutes: The import module searches for hidden rename and move operations. This intervall narrows the search. Only those file removings and addings are considered whose execution dates differ at most "maxTimeIntervalInMinutes" minutes.
  - maxFileContentDifferenceInPercent: Only those files or folders are considered as equal that differ at most "maxFileContentDifferenceInPercent" percent.
- **scmSourceLocation**:
  - scmPerlScriptLocationAndFileName: Name of and path to the Perl script that is contained in the project scmbug_user_name_translation.
  - scmDaemonConfLocationAndFilename: Location of the SCMBug daemon configuration file. It is usually placed in "/etc/scmbug/daemon.conf".
  - relativeCvsScmBugGluePath: Relative location of the SCMBug glue in a CVS repository (default: /CVSROOT/etc/scmbug).
  - relativeSvnScmBugGluePath: Relative location of the SCMBug glue in a SVN repository (default: /hooks/etc/scmbug).

6.1.4. Import SCM data

1. Checkout the project into the folder specified in the parameter "relativeWorkingDirectoryCheckoutPath" of the settings file.
2. The import of all calculation data from CVS or Subversion is implemented as command line tool: **ScmImport [cvs | svn] path\to\settings.xml**. As parameter it requires the type of the used software configuration management system, and the path to the folder with the files "settings.xml" and "logging.properties". All other settings are stored in the file "settings.xml". The tool is contained in BugzillaMetrics.jar which can be found in the folder "BugzillaMetricsFrontend\WEB-INF\lib" of the file release. Here is an example on how to start the import process. Please make sure to include the required libraries on the classpath

   ```
   java -classpath BugzillaMetrics.jar:jdom.jar:log4j.jar;mysql-connector-java-3.1.13-bin.jar
   org.bugzillametrics.scm.dataImport.ScmImport cvs /home/someUser/scmImport
   ```

6.2. Metric specification involving the SCM - Integration

The syntax and semantics of the metric specification presented in Chapter 3, *Metric Specification* does not change in specifications for SCM data. The integration merely extends the possibilities by adding SCM specific BaseFilter, Events and Weights. They can be mixed with those elements that are known from Chapter 3, *Metric Specification* . The following sections will give an overview over additional Section 6.1, “SCM - BaseFilter” , SCM - Events and SCM - Weights .

6.1. SCM - BaseFilter

Tag	(XML Element: <tag />) In this filter the user has the opportunity to select cases by tag name. It filters all cases that affect files or folders labeled with the given tag.
File type	(XML Element: <fileType />) In this filter the user has the opportunity to select cases by file type. It filters all cases that affect files with given file type.
File	(XML Element: <file />) In this filter the user has the opportunity to select cases by file name. It filters all cases that affect files with given name.
Bugzilla Developer	(XML Element: <bugzillaDeveloper />) In this filter the user has the opportunity to select cases by developer name. It filters all cases where a developer is active, who uses the given name under Bugzilla.
SCM Developer	(XML Element: <scmDeveloper />) In this filter the user has the opportunity to select cases by developer name. It filters all cases where a developer is active, who uses the given name under SCM.
Branch	(XML Element: <branch />) In this filter the user has the opportunity to select cases by branch name. It filters all cases that affect files or folders that are located in the given branch.
Folder	(XML Element: <folder />) In this filter the user has the opportunity to select cases by folder name. It filters all cases that affect folders with given name.

6.2. SCM - Events

Branch creation	(XML Element: <createBranch />) All branches that have been created within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.
Branch removing	(XML Element: <removeBranch />)

All branches that have been removed within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

File changing
(XML Element: <fileChange />)

All files that have been changed within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation. Changing means only that there is a change within the file. For changes to the path or filename use fileMove or fileRename.

File creation
(XML Element: <fileAdd />)

All files that have been added to version control within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

File moving
(XML Element: <fileMove />)

All files that have been moved to a new folder within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

File removing
(XML Element: <fileRemove />)

All files that have been removed from version control within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

File renaming
(XML Element: <fileRename />)

All files that have been renamed within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

Folder changing
(XML Element: <folderChange />)

All folders that have been changed within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation. Changing means only that there is a change within the folder. This is the case if a file or subfolder was added to or removed from the folder.For changes to the path or foldername use folderMove or folderRename.

Folder creation
(XML Element: <folderAdd />)

All folders that have been added to version control within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation. Note: CVS does not support version control for folder. For this SCM - System the import algorithm labels a folder as added when the first file or subfolder was moved into it.

Folder moving	(XML Element: <folderMove />) All subfolders that have been moved to a new folder within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.
Folder removing	(XML Element: <folderRemove />) All folders that have been removed from version control within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.
Revision added	(XML Element: <revision />) All revisions that have been added in version control within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.
Tag creation	(XML Element: <createTag />) All Tags that have been created within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.
Tag removing	(XML Element: <removeTag />) All Tags that have been removed within the time interval and are referenced in a case that matches the BaseFilter will be considered in the evaluation.

6.3. SCM - Weights

Added lines of code	(XML Element: <addedLoc />) This Weight is enabled for all Events that describe file activities. It returns for each event the number of added codelines of the file that is affected by the event. The number refers to the impact of the event.
Changed lines of code	(XML Element: <changedLoc />) This Weight is enabled for all Events that describe file activities. It returns for each event the number of changed codelines of the file that is affected by the event. The number refers to the impact of the event.
Deleted lines of code	(XML Element: <deletedLoc />) This Weight is enabled for all Events that describe file activities. It returns for each event the number of deleted codelines of the file that is affected by the event. The number refers to the impact of the event.

Lines of code	(XML Element: <loc />) This Weight is enabled for all Events that describe file activities. It returns for each event the number of codelines of the file that is affected by the event. The number refers to the impact of the event.
Number of branch events	(XML Element: <numberOfBranchEvents />) This Weight is enabled for all revision - and branch events. It returns for each event the number of subevents refering to a branch. For revision events it returns the number of branch events that happened between the given revision and the previous revision. For all branch events it returns 1.
Number of developers	(XML Element: <numberOfDevelopers />) This Weight is enabled for all revision - and file events. It returns for each event the number of developers who are affected to the file. For revision events it returns 1. For all file events it returns the number of developers contributing at least one codeline to the file.
Number of file events	(XML Element: <numberOfFileEvents />) This Weight is enabled for all revision - and file events. It returns for each event the number of subevents refering to a file. For revision events it returns the number of file events that happened between the given revision and the previous revision. For all file events it returns 1.
Number of files	(XML Element: <numberOfFiles />) This Weight is enabled for all file - and folder events. It returns for each event the number of files that are affected. For folder events it returns the number of files that are located in the folder after the event happened. For all file events it returns 1.
Number of file types	(XML Element: <numberOfFiletypes />) This Weight is enabled for all file - and folder events. It returns for each event the number of file types that are affected. For folder events it returns the number of file types that are located in the folder after the event happened. For all file events it returns 1.
Number of folder events	(XML Element: <numberOfFolderEvents />) This Weight is enabled for all revision - and folder events. It returns for each event the number of subevents refering to

	a folder. For revision events it returns the number of folder events that happened between the given revision and the previous revision. For all folder events it returns 1.
number of subfolders	(XML Element: <numberOfSubfolders />) This Weight is enabled for all folder events. It returns for each event the number of subfolders that are located in the folder after the event happened.
number of tag events	(XML Element: <numberOfTagEvents />) This Weight is enabled for all revision - and tag events. It returns for each event the number of subevents refering to a tag. For revision events it returns the number of tag events that happened between the given revision and the previous revision. For all tag events it returns 1.

Chapter 7. Test-Framework

Two test frameworks are available for BugzillaMetrics in order to check the following program functions:

1. BugzillaTest : Allows the verification of the results of metric specifications, based on the given content of a database.

2. ImportTest : Allows the verification of the correct import of data from SCM systems (CVS or Subversion). It generates real SCM systems, runs the import and compares it to expected content of the scm database.

7.1. BugzillaTest

BugzillaTests support tests of the BugzillaMetrics core. Optionally the SCM integration can be tested, too. One Test case consists of three files.

1. BugzillaTest: setup : One of the files specifies the contents of the Bugzilla database. If the SCM integration is part of the review the content of the SCM database must also be indicated.

2. BugzillaTest: metric : The second file is a Metric specification. It will be processed by BugzillaMetrics based on the database content.

3. BugzillaTest: result : The expected metric result is specified in a third file.

7.1.1. Setup

The content of the database is specified in a XML file. It consists of a root-element <db /> and two child elements <bugzilladb /> and <scmdb />. Both include a child element for each table entry in the Bugzilla- or SCM-database. The name of an element corresponds to the target table. For each column of the table there is an attribute named by the corresponding column.

7.1.1.1. Bugzilla-database:

All tables and columns can be found in http://www.ravenbrook.com/project/p4dti/tool/cgi/bugzilla-schema/ [http://www.ravenbrook.com/project/p4dti/tool/cgi/bugzilla-schema/].

7.1.1.2. SCM database:

The following items are available to fill the SCM database.

Table 7.1. Elements for the specification of database table entries (SCM objects)

Table name	**Attribute**	**Value**
<branch />	ID	Integer
	Name	String
	Enabled	True = 1, False = 0
<developer />	ID	Integer
	Bugzilla_Name	String
	Scm_Name	String
<scm_File />	ID	Integer
	Name	String
	Type	String
	FolderID	Integer
	Enabled	True = 1, False = 0
<scm_Folder />	ID	Integer
	DevelopmentLineType_ID	Integer
	DevelopmentLine_ID	Integer
	Name	String
	Path	String
	Project_Name	String
	Enabled	True = 1, False = 0
<tag />	ID	Integer
	Name	String
	Enabled	True = 1, False = 0

Table 7.2. Elements for the specification of database table entries (Activities)

Table name	**Attribute**	**Value**
<branchActivity />	ID	Integer
	Type_ID	Integer
	SCM_Branch_ID	Integer
<fileActivity />	ID	Integer
	Type_ID	Integer
	SCM_File_ID	Integer
	File_LOC	Integer
	Contributor_ID	Integer
	Contributor_LOC	Integer
	Committer_Added_LOC	Integer
	Committer_Deleted_LOC	Integer
	Committer_Changed_LOC	Integer
	Old_File_ID	Integer
<folderActivity />	ID	Integer
	Type_ID	Integer
	SCM_Folder_ID	Integer
	File_COUNT	Integer
	Filetype_COUNT	Integer
	Subfolder_COUNT	Integer
	Old_Folder_ID	Integer
<scm_Revision_Activity_-History />	ID	Integer
	Revision_ID	Integer
	Developer_ID	Integer
	Activity_ID	Integer
	Activity_Type_ID	Integer
	Log_Comment	String
	TimeStamp	YYYY-MM-DD HH:MM:SS
	Bug_ID	Integer
<tagActivity />	ID	Integer
	Type_ID	Integer
	SCM_Tag_ID	Integer

7.1.2. Metric specification

The test framework uses metric specifications of the normal syntax. It offers the full range of functions.

7.1.3. Result comparison

The result of the metric calculation, which is a time series, is an XML file. The root element is <metricResult />. Its children are the specified groups: <group name="...">. For each of the calculated values of the time series, there is an element <timePeriod scope="..."> in which the calculations are stored: <calculation name="sum"> "value" </ calculation>.

7.1.4. BugzillaTest example

Database content:

```
<db><bugzilladb>
  <user userid="1" />
  <user userid="2" />
  <product id="1" />
  <component id="1" />
  <bug bug_id="1" creation_ts="2006-08-07 12:00:00"
    priority="P4" assigned_to="2" />
  <bug bug_id="2" creation_ts="2006-08-14 12:00:00"
    priority="P1" />
  <bug bug_id="3" creation_ts="2006-08-21 12:00:00"
    priority="P2" assigned_to="2" />
  <bug bug_id="4" creation_ts="2006-08-28 12:00:00"
    priority="P3" />
</bugzilladb></db>
```

Metric specification:

```
<?xml version="1.0" encoding="UTF-8"?>
<metric>
  <baseFilter>
    <value field="assignee">1</value>
  </baseFilter>
  <groupingParameters>
    <none />
  </groupingParameters>
  <groupEvaluations>
    <calculation name="sum" >
      <sum caseValueCalculator="default" />
    </calculation>
  </groupEvaluations>
  <caseValueCalculators>
    <countEvents id="default" >
      <event>
        <endOfTimeInterval/>
      </event>
      <weight>
        <mapping field="priority">
          <map from="P1" to="4" />
          <map from="P2" to="3" />
          <map from="P3" to="2" />
          <map from="P4" to="1" />
        </mapping>
      </weight>
    </countEvents>
  </caseValueCalculators>
  <evaluationTimePeriod>
    <timePeriod>
      <start>2006-08-14</start>
      <end>2006-08-27</end>
    </timePeriod>
  </evaluationTimePeriod>
  <timePeriodGranularity>
    <week />
  </timePeriodGranularity>
  <fixedFields />
</metric>
```

Expected result:

```
<metricResult>
  <group name="none">
    <timePeriod scope="week 33/2006">
      <calculation name="sum">4</calculation>
    </timePeriod>
    <timePeriod scope="week 34/2006">
      <calculation name="sum">4</calculation>
    </timePeriod>
  </group>
</metricResult>
```

7.2. Import-Test

The test of the data import is realized by ImportTest. The framework generates real SCM systems that have to be specified during setup. The result content of the database written by the import module is compared with a mysqldump report. One test consists of two files:

1. ImportTest: setup : scm specification.

2. ImportTest: result : mysqldump report.

7.2.1. Test setup

The information for the SCM initialization includes the following main elements that are integrated in a XML schema:

1. Content of the Bugzilla database.

2. Information for the SCM scenario (CVS (NT) and / or subversion):

   The SCM scenario is defined for a specific version management system. There are elements <cvs /> and <svn /> for CVS and SVN. Within these elements all activities in the SCM-repository can be defined. These activities must be written in the desired order of execution. An exception is <import />. This element may be no more than once used and will always be the first executed operation. In the specification of a CVS repository, the import operation must be used. Otherwise CVS does not initialize the repository and the following operations can not be executed. The import element, however, can also be empty. Optionally, the execution date of an operation can be indicated: time = "yyyy-mm-dd hh: mm: ss"

   The time should only be defined in test cases, which depend on a time specification. Otherwise, no time should be defined. A specification could cause the following problems:

   a. A specification causes the change of the system time. This operation requires administrative rights.

   b. The development environment could crash (eg Eclipse).

   c. The filesystem could become inconsistent (eg Linux ext3 filesystem).

   The specified configuration of the repository is independent of the version management system. So if the same test for CVS and SVN should be checked, the configuration has to be written only once. Both scenarios can be defined in a common file. The tests will be executed one after another and will be compared with the same expected result.

The following operations can be used for the SCM setup:

- Add operations: Operation will not be committed!

  1. <add SourceLocation="" sourceFileName="" targetLocation="" targetFileName="" time="" />

  2. <addFolder Location="" time="" />

  3. <branch BranchName="" comment="" time=""/>

  4. <tag Location="" tagName="" comment="" time=""/>

- Checkout operation: Checking out a project branch or tag.

1. <checkout BranchName="" projectName="" time="" />

- Commit operation: Committing previous executed operations.

  1. <commit Comment="" time="" />

- Import operation: To import files directly after the initialization.

  1. <import Comment="" time="">

- Add file after initialisation: Child element of <import />. Description of a file that should be imported after initialisation of the test repository.

  1. <file SourceFileName="" sourceLocation="" targetLocation="" targetFileName="" />

- Move operations: Moving a file / folder in the local working copy. Operation will not be committed!

  1. <moveFile FileName="" sourceLocation="" targetLocation="" time=""/>

  2. <moveFolder SourceLocation="" targetLocation="" comment="" time=""/>

- Remove operations: Deleting a branch, file, folder or tag in the local working copy. Operation will not be committed!

  1. <removeBranch BranchName="" comment="" time=""/>

  2. <removeFile Location="" fileName="" comment="" time=""/>

  3. <removeFolder Location="" comment="" time=""/>

  4. <removeTag TagName="" comment="" time=""/>

- Rename operation: Renaming a file / folder in the local working copy. Operation will not be committed!

  1. <rename Location="" oldName="" newName="" comment=""/>

- File replace operation: Replace a file with another. Used to simulate a file change.

  1. <replaceFile SourceLocation="" sourceFileName="" targetLocation="" targetFileName="" time="" />

7.2.2. Result comparison

The Import test launches an import process based on a SCM system with the above defined scenario. The result will be compared at database level. This is done using a database export from mysqldump as XML file. Because of differences depending on time and executing developer, the following entries are ignored:

1. Records of the type TimeStamp: Values depend on test execution time.

2. Revision_ID in SCM_Revision_Activity_History table: This value is not predictable because CVS generates an ID for each commit.

3. Developer names in the Developer table. The name corresponds to the user logged on the operating system.

The comparison is more flexible than a simple file-diff, but it has disadvantages:

1. A structural modification of the database means that all the tests fail. This does not apply to an expansion with new columns.

2. Two tables that contain the same records but in different order are not equal.

7.2.3. ImportTest example

Example: SCM test case for committing TestFile1.java in root folder of trunk (SVN repository):

Setup:

```
      <scm>
        <bugzilladb>
        </bugzilladb>
        <svn>
          <add sourceLocation=""
               sourceFileName="TestFile1.java"
               targetLocation="trunk"
               targetFileName="TestFile1.java" />
          <commit comment="bug1: Kommentar fuer den commit" />
        </svn>
      </scm>
```

Result comparison:

```
      <?xml version="1.0"?>
        <mysqldump xmlns:xsi=
          "http://www.w3.org/2001/XMLSchema-instance">
        <database name="scm">
       <table_structure name="Developer">
            (...)
       </table_structure>
       <table_data name="Developer">
         <row>
        <field name="ID">1</field>
        <field name="Bugzilla_Name">Error</field>
        <field name="Scm_Name">root</field>
         </row>
       </table_data>
```

```
        (...)
      </database>
    </mysqldump>
```