Contribution ID: **83** Type: **not specified**

Common framework for extracting information and metrics from multiple change trackers

Thursday, 29 March 2012 11:20 (20 minutes)

Description of the Work

The first step towards producing a uniform framework for dealing with metrics and KPIs, was the clear definition of the common parameters that was then successfully exported by each middleware. These parameters form the inputs for any metric engine. It was imperative that metrics receive exactly the same form of data from multiple institutions and produce outputs in a form that is relevant to each individual customer.

It became apparent early on in the project that a uniformly extensible framework for dealing with inputs from each middleware product teams change tracker (i.e bug/feature tracker) could easily be achieved using an XML based format starting with a lightly constrained XML schema and progressing to a very tightly controlled schema.

Once all input data were defined for the QA group, it became apparent that there was a fundamental issue of mapping each middlewares change tracker category name to an individual product. This required a specific XML mapping description to allow the QA group to ascertain how each tracker maps its changes to each individual product.

With the mapping XML and change tracker XML in place this paved the way for defining a Java based framework for producing automated daily or periodic charts, producing metrics and KPIs for each of the customers within EMI.

Another very useful requirement of many customers was a dashboard showing tabulated data obtained from the change tracker XML. This was built with a back-end query engine giving various views of the data to many customers. It also produces datasets that the user can easily plot.

In more recent months it was decided to strip the Java plotting framework back to a minimal set of operations, so that specially tailored charts could be generated for any customer who requests them. This has proved invaluable from the point of view of being able to corroborate the KPIs of customers and to give quick turnaround times on producing new statistics.

Conclusions

The production of a common XML format for change management avoided the painful migration of each bug/feature tracker to a new system where each middleware had little or no expertise.

All middlewares found it possible to produce and export the common XML format. Since it was simple in structure, it was relatively straightforward to wrap with a query engine to produce a dynamic dashboard.

The Java Framework uses the common change tracker XML and mapping XML to produce many different KPIs in the context of a product. Without these mappings this would be next to impossible, requiring manual queries on each bug-tracker, correlation of data, and manual plotting of KPIs for every milestone or deliverable.

Without a general framework in place, consulting the change trackers for more than 50 Products would result in wasted effort by developers in approximately 30 Product Teams. It makes much more sense to control the flow of information like this through the QA group.

Impact

In all, so far, six different bug-trackers have been integrated into the common XML based RfC tracker.

Currently the RfC dashboard is used to verify the correctness of the RfC report produced by the tools group (SA2.4) for the executive management team (EMT) weekly meeting. SA1 use it to export data which is then used as input data to produce their quarterly reports. The metrics group (SA2.3) use it internally to verify whether each element of an RfC has valid values. The Quality Control team use it to correlate which products produce valid tests at the time of each release update and major release.

In year one of EMI, the development of the Java based framework made it possible to automate the production of a weekly PDF report of all immediate and high priority tickets in each of their transition states, for given time periods, for varying defect/feature categories and ascertain whether they arose from production, development or testing.

In year two of EMI, it became increasingly important for the Quality Control (QC) team to see which requests for change (RfCs) due for realise included regression tests in the case of defects and functional tests in the case of feature requests. This feature was slowly integrated into the change management XML format without any disruption to existing customers. Once fully integrated it was heavily constrained.

Visually speaking, the RfC dashboard produces columns of data such as priorities, severities and detection areas for each RfC, details about each individual ticket and links back to the originating GGUS ticket if it exists. The output of the dashboard is a table of subsets, presented as Google Apps based tables or data sets that can be imported by customers into Excel so that they can produce their own plots.

URL

http://emiqa.web.cern.ch/emiqa/reports/latest/
http://lxbra1902.cern.ch/EMI/BugListing

Overview (For the conference guide)

An important aspect of EMI is the delivery of 'quality software'. For this reason the quality assurance (QA) group was introduced. There are a key number of beneficiaries of this work in a number of work packages. These EMI development and support activities are required to produce key performance indicators (KPIs) and metrics for milestone, quarterly and yearly deliverables based on the information provided by the QA group. However, EMI has a large number of varying sized products, different middlewares and various bug/feature request for change (RfC) trackers used by each product team. The only way to reliably produce KPIs and metrics related to change management in such a varied project is to introduce simplifying, common environments that are readily accessible by varying numbers of customers of the project. For this reason an extensible XML-based framework was generated for storing, plotting, querying and tabulating change tracker information for all its customers.

Primary author: Dr KENNY, Eamonn (TCD)

Presenter: Dr KENNY, Eamonn (TCD)

Session Classification: EMI: Software Quality Assurance