ISM’s Predictive Pro predicts potential code weaknesses

Code analysis tool identifies bug hot spots to focus QA efforts

By Rick Grehan

June 15, 2006

Most QA tools manage project and bug tracking. They largely manipulate and organize information gathered in the past, and in so doing, tell you where you are on a project. If they step into the future, they only do so to assist in constructing timetables of proposed activities such as task planning.

Now comes a tool that processes data harvested from the past to inform the future: ISM (Integrated Software Metrics) Predictive Suite -- QA tools that act as “AI bug prophets.”

Behold the future
Predictive’s crystal ball is a combination code-metrics engine and artificial-intelligence system. The former reads through the source code, calculating and gathering a number of code metrics on each method. The latter analyzes the metrics and makes intelligent prognostications of which methods likely will be the source of future errors.

Predictive does not locate bugs; it locates -- or, rather, predicts -- the location of bugs. More precisely, it predicts which methods within a project are likely sites for future bugs: In the parlance of the documentation, Predictive “identifies regions of risk” in an application. The application’s source code can be C, C++, C#, or Java.

So, Predictive alerts developers and QA engineers to those methods that are likely to be the breeding grounds for future bugs. As a corollary, developers and QA can intelligently prioritize their efforts and don’t have to waste their time overly testing areas of their application identified as low-risk.

There are three members of the Predictive family: Lite, Pro, and Server. Lite is suitable for single users working on small to medium projects -- although there is no limitation to the size of the project that Lite can handle. Whereas the Pro and Server versions of the Predictive suite use AI to learn the nuances of your project team’s coding weaknesses, the Lite edition incorporates a “pre-taught” AI brain. That is, Lite’s predictive abilities are based on what it learned by examining a wide selection of standard software fed to it before it left its birthplace.

I tested the Pro version, which sports a 98 percent accuracy rate according to ISM. (The company pegs Lite’s accuracy at 82 percent, which isn’t bad for a pre-built AI brain.) Pro is meant for larger projects; Predictive Server edition goes a step further, targeting multiple-project enterprises and delivers browser access into an engine with the same capabilities as Predictive Pro. Server also maintains an archive of past bug data for all projects in your organization and consults them as necessary when making its prophecies or delivering its error-trend reports.

As your project’s development moves along and bugs are entered into your favorite bug-tracking tool, that information must be passed into Pro: The more information Pro receives, the more it learns and the better its accuracy.

Pro does not integrate directly with your bug trackers; it imports data only in CSV (comma-separated value) form, and that’s something of a limitation. Being able to import data in Excel, Access, or SQL Server formats, for example, would be nice.

The Pro edition also comes with an Error Trend view, which uses historical information to show you the methods in your project that are suffering from chronic bug infestation. It also allows you to verify how close the predictions are to reality.

Prediction in practice
Predictive Pro’s major functions are categorized as “modes.” Standard Mode uses the same non-AI forecasting used by the Lite edition. Standard Mode is made available in the Pro edition for the simple reason that it provides results fresh from the box, with no learning phase needed.

Heuristic Mode employs the educable AI engine and is the mode you’ll use when a project is under way. Error Trend Mode, as mentioned above, highlights methods that have proved to be bug breeding grounds so you can avoid them.

All three modes present in a spreadsheet-style tabular view that’s easy to navigate. Each row in the table corresponds to a method in the application. A navigation window to the left allows you to drill down into methods within specific modules or to rise to the top of the project tree and see all the methods. Columns in the table hold the metric information and historical bug data for each method.

Predictive Pro collects quite a few metrics, including simple information such as lines of code, characters of comment, lines of comment, and so on. It also calculates more elaborate metrics, such as number of returns in a method (the fewer, the better); cyclomatic complexity, a measure of the branching within flows of control in the method; Halstead metrics, 11 measurements that estimate the mental effort required to develop the method; and structural and architectural complexity. Structural measures the complexity of the code based on its configuration, and architectural measures the complexity of the code based on its interaction with other code. Both are metrics unique to the Predictive Products.

Cells in the table are colored green, yellow, and red, based on the method’s risk level. The specific metrics -- or historical bug data -- used in the analysis are also color-coded. That is, if a method has been flagged as high risk, metrics that have exceeded acceptable limits show up in red -- making it easy to identify them during code review.

Minimal bugs
As with all AI-based products, Predictive Pro is no panacea -- even when it works at its best. To make the best use of it, you must be well-acquainted with your project’s architecture. For example, even if the Error Trend report indicates that a particular method is error-prone, it may actually be a little-used method that has less of an impact on your product than other, less error-prone methods.

Although it’s unlikely that developers of small projects will need Predictive Pro, large projects pressed for time and resources should find this a worthwhile guide in directing testing efforts. Used over time, it will get better and better at spotlighting specific weaknesses in your team’s development efforts -- and may just save you a boatload of time and frustration.

Print this