

Development Information Systems - A New Paradigm for Software Development

In an interview with CIO Magazine, Dwight Schettler, OpenView CPE Operations Manager, Hewlett - Packard Company, added: "We had been using a whole host of homegrown, in combination with some off the shelf, tools, but the capability that comes along with the DISCOVER Development Information System is certainly something new. The sum of all those parts does not equate to the feature set of something like DISCOVER. I think the industry is beginning to catch up and see the value in it."

INTRODUCTION

The emergence of new computing capabilities, including languages, processing capabilities and communications increases software complexity. As a result, software becomes increasingly more difficult to preserve and maintenance costs spiral out of control. (Figure 1)



Figure 1. The emergence of new computing capabilities, including languages, processing capabilities and communications increases software complexity. As a result, software becomes increasingly more difficult to preserve and maintenance costs spiral out of control.

What was once well-structured, intuitive code now assumes the consistency and organizational characteristics of a potato left in a dark, damp basement too long; moldy with all kinds of unknown things growing out of it. According to Dan Stang, a research analyst at Datapro, as much as 75 to 85 percent of software development is spent on reengineering. As a result, new product development is left with minimal time and budget resources while its value and growth stagnate.

To compound an organization's software problems, development's delivery schedule is often unpredictable and at the mercy of individual developers. Developers tend to work in seclusion and are often forced to fend for themselves as opposed to acting as integrated team members. In addition, code is written by one, and rewritten by another; defects are fixed, only to have a new problems develop in unrelated areas;

and development cycles are extended, increasing test cycles all while scarce resources are wasted. If objections are raised, developers merely move on to the next position or project in the next company. Management has little or no control and many processes are manual in nature. The entire process is out of control!

These problems are common in most software development environments. The key question is - What is the solution?

Enter a new paradigm - a new class of software development applications - the Development Information System. Modeled after other successful information systems, from manufacturing (ERP and MRP) to human resources (HRIS) and finance, Development Information Systems are designed to provide all software developers and their managers with universal access to a one inclusive source of information.

For years executives have been investing in information systems for other areas of their business, often paying hundreds of thousands, even millions of dollars, to secure the management of their key assets.

The time has come for software development to take advantage of the same theory and practice that it has created for its corporate siblings.

THE INFORMATION MODEL - CONTROLLING THE ASSET

At the heart of any information system is a central database that stores code entities, as well as capturing and recording their interrelationships. This "Information Model" provides management with the fundamental capability to secure, catalog and codify its assets, in DIS's case, the asset is software source code. (Figure 2)

Consistent with established software development processes (SEI, ISO9000, and Six Sigma), the Development Information System's initial step is to gain control of the software asset. In order for a DIS to be effective, all elements of the software must be registered in a database and the relationships between these elements must be captured. Source code, specifications, tests, documentation -- literally everything

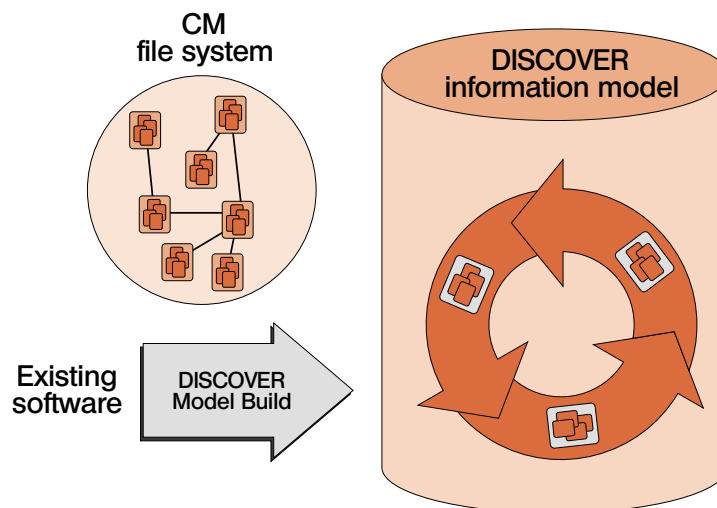


Figure 2. At the heart of any information system is a central database that stores code entities, as well as capturing and recording their interrelationships. This "Information Model" provides management with the fundamental capability to secure, catalog and codify its assets, in DIS's case, the asset is software source code.

associated with the software -- must be included. If an element is excluded, the potential effectiveness of the system is jeopardized.

Once the database is built, every information system essentially performs three basic functions:

- 1 The ability to query or navigate the data;
- 2 The ability to automate change; and
- 3 The ability to generate reports and establish control of assets.

These three functions are the hallmark of "integrated" applications.

In the case of ERP these functions are the bill of material, purchasing and master scheduling. Each application works off of the central database in a coordinated manner, and is able to take advantage of information that once was only privy to its creators and immediate users.

Much like ERP, DIS uses common data and tailored applications to integrate disparate pieces and groups of information. The DIS uses this information to query source code, perform analysis of existing code and proposed changes, perform code reviews, optimize tests, automate documentation, component mining, packaging, and generate reports.

QUERY - COMPREHENDING THE ASSET

One of the most valuable functions a DIS provides is the ability to efficiently and accurately query its information model. Development Information Systems understand the relationships between individual parts of source code as well the location of related information, which facilitates its sophisticated query abilities. In addition, the query function of the DIS can provide detailed and "consumable" data both in textual and graphical formats. These familiar, descriptive reports represent the relationships between source code and documentation, tests, specifications and even software artifacts like defects.

In conventional software systems, the process of identifying a portion of the code and its related support systems can be a challenge, even for the most experienced developers. For those developers who are new to the team, the task is essentially impossible. The DIS's ability to perform queries helps bring new employees up-to-speed quickly and allows veterans to efficiently navigate the system.

In addition to its query functionality, the DIS's heightened level of information access allows analysis to be performed, which extends the depth and value of source code comprehension. For example, the potential impact of a code change on software can be explored prior to the change, thus eliminating the "fix 1, break 2," syndrome of bug fixing. Also, dormant code and complex header structures can be identified in an effort to unwind the code's complexity. Portability, globalization and adherence to programming standards can also be explored.

CHANGE AUTOMATION - IMPROVING THE ASSET

Once a Development Information System establishes detailed source code comprehension, initiating change can be a less daunting and more manageable process. The computer can automate code changes for en-mass resolution, or more fastidious piecemeal operations, while providing the developer with an understanding of all impacts and necessary resolutions. A DIS can extract dormant code and simplify complex header files, because it has the intelligence to recognize the relationships between the change and its context.

The DIS's information model automatically identifies the portions of the code structures affected by changes and suggests the tests that need to be run. This insight significantly reduces test cycles and can easily update software documentation. All applications utilize the speed and accuracy of the computer while benefiting from an intrinsic understanding of key relationships captured by the DIS database.

REPORTING AND CONTROL - MANAGING THE ASSET

The ability to generate reports represents management's degree of asset control and accountability.

In the past, software R&D managers did not have the ability to quantify or assess the status of their efforts. They were the only managers within the organization who lacked these important productivity assessors, and as a result, they were the only department that could not accurately predict or control business-affecting deliverables.

Generating reports for productivity and status assessment within the DIS is simply matter of documenting specific queries or analysis. Data on any quantifiable attribute of the information model, like its total number of lines of code, or "Meyers complexity," are readily available. The DIS is also able to provide complex reports such as the quantification of internal code quality as measured against company programming standards. These reports provide management with a method to track code quality prior to testing, thus enhancing their ability to provide accurate delivery timeframes.

In addition, the DIS can process reports on the number of defects in selected subsections of code, which management can use to expedite remediation. For example, defect reports help management know whether or not it is more efficient to rewrite or fix the offending body.

For the first time, software development managers have an information system that provides them with the tools to assess the state of their products, and to make educated estimates regarding time-to-market, code quality and employee productivity. (Figure 3)

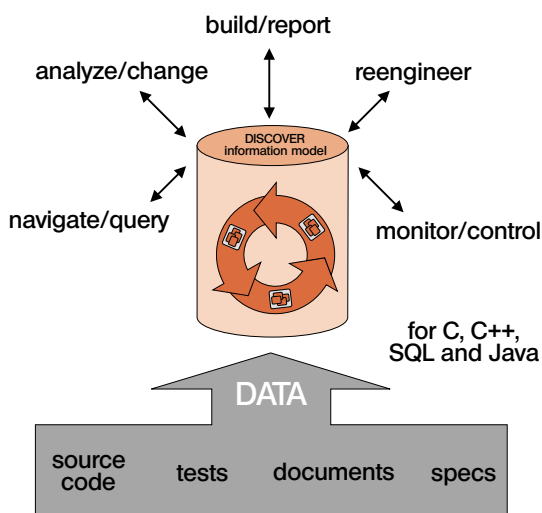


Figure 3. For the first time, software development managers have an information system that provides them with the tools to assess the state of their products, and to make educated estimates regarding time-to-market, code quality and employee productivity.

CONCLUSION

DIS prevents duplicated development; it negates the corruption of new functions by the resolution of prior errors; it supplants manual code reviews and ineffective tests and all resulting errors. But perhaps DIS's greatest benefit is derived from its impact on software quality.

Can anyone place a price on lowered defect rates and solutions to previously unsolvable issues? The reduction of maintenance costs alone can be significant, and more importantly, can allow for the shift of resources from maintenance tasks to new product development. If even 10 percent of the current time spent on maintenance could be shifted to new product development, the impact on time-to-market and new feature creation would be tremendous.

The concept of information systems is not new. However, harnessing the power of information systems technology for software development is revolutionary, and promises to turn the industry on its ear.

The very nature of software Development Information Systems - the connectivity it provides to developers and the accountability it provides to management-will help information system executives realize the power and value in both the development process and the software systems themselves.

DIS is a new paradigm for software development. And frankly, it's about time that the software development industry enjoys the same capabilities, and reaps the same rewards, that it has provided to its customers for decades ■

Bruce Boes is Vice President of Marketing at UPSRING Software, Inc. (Burlington, Mass.). UPSRING is a software infrastructure solutions company that helps eBusiness and other mission-critical development organizations to analyze and evolve their software infrastructure by providing solutions that address Productivity, Quality, and Change . UPSRING develops, markets, and supports two compatible product lines; CodeRover, standalone targeted applications to enhance personal productivity of software developers, and DISCOVER, an enterprise-wide solution for organizational productivity.