Pulling together

Improving engineering management though collaboration. ***By Dave Robertson.***

Engineering is multidisciplinary and can therefore benefit from 'consolidated thinking'. However, due to the dissimilar nature of the various disciplines, concurrent engineering has, historically, precluded any accessible method for documenting and coordinating results. Until recently, the effort involved in combining them has rendered true concurrent engineering almost impossible to achieve.

But with the advent of computer aided engineering, the development landscape has evolved into a more homogenous environment. It is commonplace to find engineers with contrasting skills working side by side on the same project using the same workstations, albeit using different software design and development tools. Under these circumstances, concurrent engineering has taken a giant step forward and looks to become a real possibility in improving the way we work.

This is particularly important for modern product development projects, which involve the creation and evolution of large amounts of digital data and efficient methods for data management.

A consumer or enterprise product consists of many elements – such as source code, marketing collateral, digital assets, hardware designs, configuration and test scripts. The use of a single repository for files from multiple sources makes sense and a Software Configuration Management (SCM) system meets that requirement with change history, and more, as a bonus.

However, in the world of engineering management, if the SCM tool takes only a software developer-centric view of the world, non software developers can find using it a very daunting prospect. This may reinforce their compartmentalised view that SCM tools are only for code, documentation management systems are for documents, digital asset management systems are for graphics and so on.

> *"It is commonplace to find engineers with contrasting skills working side by side on the same project"* ***Dave Robertson****, Perforce Software*

In reality, there are many SCM tools that support a range of contributors – including partners and third parties – in different parts of the world and that enable them to work well together. This leads to a reduction in the administrative overhead of managing separate systems for each type of file and reduces the inherent risk of integrating disparate systems. In addition, bringing the whole production effort under one system helps to manage complexity, reduce development time and reduce costs. It gives high visibility of any changes to everyone in the design process, so that development decisions are based on consistent and up to date information.

Using an SCM tool solves the problems associated with having multiple versions of a file and different people working on the file at the same time making many different changes. If the saving and repository scheme is fast and simple, all the changes are incorporated automatically and the conflicts are minimised. This saves hours compared to the traditional approach of bringing together different parts of the design team once a week to combine the developments in hardware, software and documentation.

The speed of change in the marketplace means developments are continuous and new versions of software are being created by various partners all the time. Embedded developers and designers grappling with variant management must understand what is

different between the software that they write for each edition of the product and what is the same.

Unless these differences are tracked effectively, developers will find themselves reproducing the same work for multiple editions of the product. Not only does that impose an unnecessary development overhead, but it also introduces more risk. Bugs may find their way into the product because the same functions are being rewritten multiple times, possibly by different team members, in different ways. It makes sense to automate the process as much as possible to allow engineers to work the way they want to.

When a developer needs to code a variant of the software for a different platform or market, the traditional method involves copying a set of source code files so that they can be worked on independently of the main code. When this happens, the system effectively copies the code into a branch, so that the developer can work on it without affecting the mainline development.

Using an SCM tool, the concept of 'lazy copying' uses pointers to manage the code base more effectively. Instead of copying a whole set of files to a new branch, the SCM software uses pointers to refer to the original files. An embedded software developer working on a new branch will see all of the files, represented logically on the system as if they were copies of the originals. The only time a new file is physically created on disk is when the developer alters it. By reducing the storage and processing overhead, this 'lazy copying' approach also brings another benefit: better performance.

An SCM system helps to automate and manage the product variants that are created as part of the collaborative process and the third party relationships; allowing outside developers to see the current state of the design and how they can fit into it without multiple engineering meetings.

The use of an SCM tool provides traceability of the development, giving confidence that, at any stage, the exact state of the design can be recreated. Archiving and managing millions of files is not a viable way to do this, so the reverse delta approach of keeping the latest version and capturing the changes that led to it, along with the timestamp, allows the state of the project and any variants to be recalled for any given time. While SCM tools provide these features by default, they also provide access control, ensuring unauthorised users are not able to move, modify or copy a file.

Although it may have been cited shamelessly for many years, concurrent development has only recently been truly realised. Concurrent development is no longer an unachievable ideology; SCM tools are a key part of the collaborative design and development process, linking internal teams across multiple disciplines and third party developers. But this has to be achieved with the minimum impact on the design flow and the company resources to ensure it is actually used by those design teams and the full benefit realised, cutting the cost and time of the whole system design.

Author profile:

Dave Robertson is vice president international for Perforce Software (**www.perforce.com**)