REPRINTED FROM: AUTOMOTIVE DESIGN & PRODUCTION MAGAZINE

Introducing Application LIFECYCLE MANAGEMENT

PLM for managing software is not PLM. It's ALM, application lifecycle management.

by Lawrence S. Gould > Contributing Editor

Blame mechatronics. No, even before that: Blame the increase in software in today's vehicles, whether firmware, application software, embedded systems, or mechatronics. "Software is quickly surpassing hardware's dominance in product development, particularly within technologically complex products and industries, such as automotive, aerospace/defense, and medical device manufacturing," according to Stefano Rizzo, senior vice president strategy and business development for Polarion Software (polarion.com).

Automakers using product lifecycle management (PLM) for managing the lifecycles of vehicles cannot be faulted for turning to these same systems for software development. It makes sense to use something PLM-like to hasten time-to-market, improve the efficiency of software development, solidify the collaboration between everyone involved in software development (including requirements specification, coding, testing, deployment, and continual software maintenance), and meet regulatory requirements.

One problem: PLM is not efficient at managing software development. Says Michael Azoff, principal analyst at the London-based Ovum Consulting (ovum.com), "In product development, software changes are far more frequent than hardware changes and keeping track of which firmware belongs to which hardware component turns into a monumental exercise in version tracking and traceability."

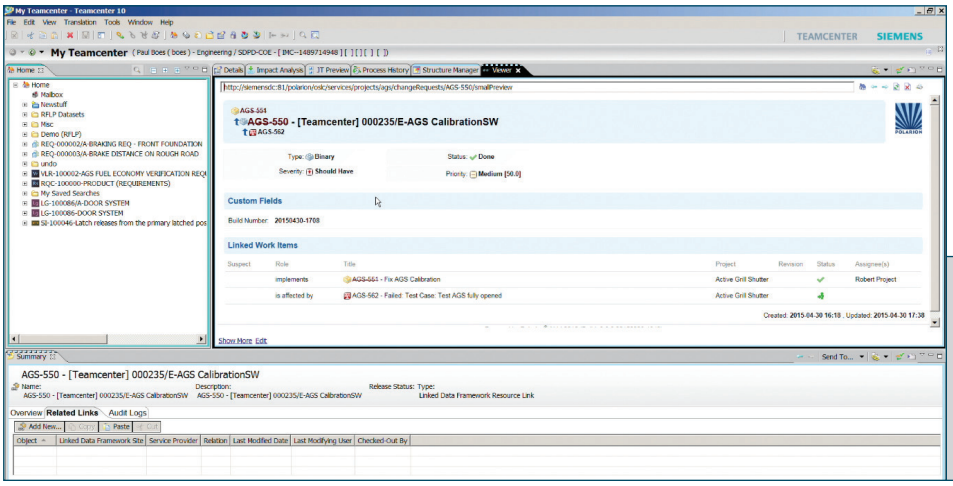
Meet ALM, application lifecycle management.

Software is different

ALM, says Azoff, is "the process by which information technology and software development organizations create, deploy, and operate software over its full lifecycle." While product design for physical things and software things is increasingly computerized, the processes are different. Product development, says Rizzo, typically follows a "cascading waterfall approach": product is *conceived* based on information from the marketplace, which is translated into customer requirements, and then into technical specifications; *designed* through a series of steps involving creation, refinement, testing, and validation; *produced* using various manufacturing methods; and then *serviced* (including repair, maintenance, and waste management), which is increasingly being considered early in product development.

Software development used to follow the waterfall approach, says Rizzo, but now it includes application project and portfolio management, project inception and requirements gathering, requirements management, design and use-case analysis, coding, testing and quality assurance, build release and deployment, and ongoing software maintenance. Software development is very much an iterative process consisting of "short, rapid `sprints' with requirements changing frequently and many ongoing revisions," says Rizzo.

Granted, "lean manufacturing" methodologies for developing physical products are similar to the "agile development" methodologies used in software development. And PLM does an excellent job at managing product-related workflows, specifications,



◀ The Polarion Connector for Team-center integrates ALM and PLM, which provides easy access to product and process data and end-to-end traceability for complex, multi-system product development.

ALM & PLM Similarities and Differences

Similarities	Differences
Both systems are built around process and core disciplines	ALM is centered on software "files" and prescribes a process to create software applications. These applications consist of multiple item types and complex relationships between these item types, which in turn create impact trees. PLM is oriented around "parts" which form a tree structure of part-of relationships.
Both systems incorporate workflow, variant management, test management, requirements, and specification management	ALM deals in the abstract. PLM deals in the concrete. In ALM, software engineers envision, elicit, define, implement, test, and maintain abstract functions. PLM's scope is to deliver a bill of materials to the production chain. The function of the components in PLM is the component itself.
Both systems allow linking components to each other	In PLM there is a "part-of" link type creating decomposition hierarchies. In ALM there are many different link types creating dependency hierarchies.
Both systems allow linking information to components	In PLM this information is pure mathematics: formulas, tolerances, diagrams, etc. In ALM the information linked to items is descriptive: textual, mock-ups, user stories, test scenarios, etc.
In both environments there is a wide usage of models	In PLM models follow the part-of decomposition and define product shapes. PLM models are usually divided into different layers representing different product subsystems: electrical layout, braking subsystem, transmission, interior, etc. In ALM a model follows the functional decomposition by means of diagrams like entity-relationship or object-oriented.

designs, and versions, but it falls apart doing the same for software. Software development is simply too complex for PLM.

Other differences exist. PLM focuses on physical, manufactured "parts." ALM focuses on software files and items (e.g., requirement document, software code, a test case) and changes to those files and items. Second, PLM is hardly browser based. ALM is, which enhances collaboration, development, test, and deployment. Says Rizzo, "It is not possible today for a user to perform a detailed 3D rendering and bill of materials with traceability links of a vehicle's transmission displayed in China from a server located in Stuttgart."

Next, PLM traceability focuses on the decomposition of a complete system; that is, a part or component as it relates to a subassembly, which relates to the finished (assembled) product. In ALM, traceability focuses on the links between files and items, even if they exist in different phases. For instance, says Rizzo, "A change to a requirement may impact a line of code, or require a new test case to be developed to validate the new requirement." A problem occurs where PLM considers software a "part," but does not consider the details in the lifecycle development of that chunk of software. This is especially true in mechatronics. "Soft-ware quality issues lie at the bottom of many costly product failures and may drive a product recall. And yet product engineers with their PLM tools lack the ability to get to the bottom of software related issues," he continues.

Mash 'em together

Mechatronics is forcing the need for both PLM and ALM, as well as the need to have these systems work together. The PLM vendors have taken notice. For example, one of the goals of the partnership between Siemens PLM Software and Polarion is the real-time synchronization of software, electrical, and mechanical development. The first product as a result became available in June: version 1 of the Polarion Connector for Teamcenter, which works with Polarion ALM 2015 and Teamcenter 10.1.4.

For $890, the connector permits integrated requirements management (such as the bidirectional referencing of software and product requirements), traceability at all levels (with no data duplication in either PLM or ALM), integrated software change management across both PLM and ALM, and closed-loop embedded systems and software. Benefits of this approach, explains Vera Sparre, Polarion's director of global marketing, include increased productivity "through closed-loop software and product development from inception to end-of-life"; better quality assurance through "better modeling and simulation as part of model-based systems engineering for continuous software validation"; better "cost containment" through "effective software delivery and reuse, as well as optimization of software design decisions in context"; and better "scalability due to the proven enterprise infrastructure that requires minimal organizational adjustment."

The integration of PLM and ALM winds up being a more-encompassing realization of what Siemens PLM calls systems driven product development (SDPD), resulting in a variety of benefits for both manufacturers and consumers. At the very least, according to Siemens PLM, ALM users no longer have to switch to PLM to search, view, and modify data residing in Teamcenter PLM, or vice versa. ■