Product Design for Variants

Considerations, incentives, and best practices

Joanne L. Scouler (jscouler@us.ibm.com)
Curriculum Architect
IBM

Martin R. Bakal (bakalm@us.ibm.com)
Worldwide Offering Manager, Electronics Industry
IBM

07 May 2013

Recognizing the need to design for product variants and the best methods to manage variants is critical to product development success over time. Variant management is how you organize the list of feature variants and put them into products. With new product development projects, planning for future variants is not typically done at the beginning. In this article. Joanne Scouler and Marty Bakal explain how to get started with variants. They illustrate their points with examples from Eaton Corporation's experience in developing variants of transmissions for heavy-duty hybrid vehicles. The authors also describe how various Rational software products can help.

Introduction

To understand when and why to design products for variants, look for the points of difference. Two different models of a truck, for example, probably have a dozen or more features that differentiate them.

Companies often don't start out designing for variants, sometimes because they lack market experience with a new product. A product development team might develop an initial product for a specific customer or use case. As the product gets customer feedback, requests for multiple variations of the initial product start to come in. If the product variants are not developed effectively, maintaining and improving each product version simultaneously can become inefficient and time-consuming because you duplicate work.

Maintaining similar software code separately across multiple products can take more development resources than starting out with product variants in mind. If you needlessly create duplicate work for each variant, eventually you cannot respond to new customer requests for variants quickly enough. A new product development process to include variant management becomes necessary.

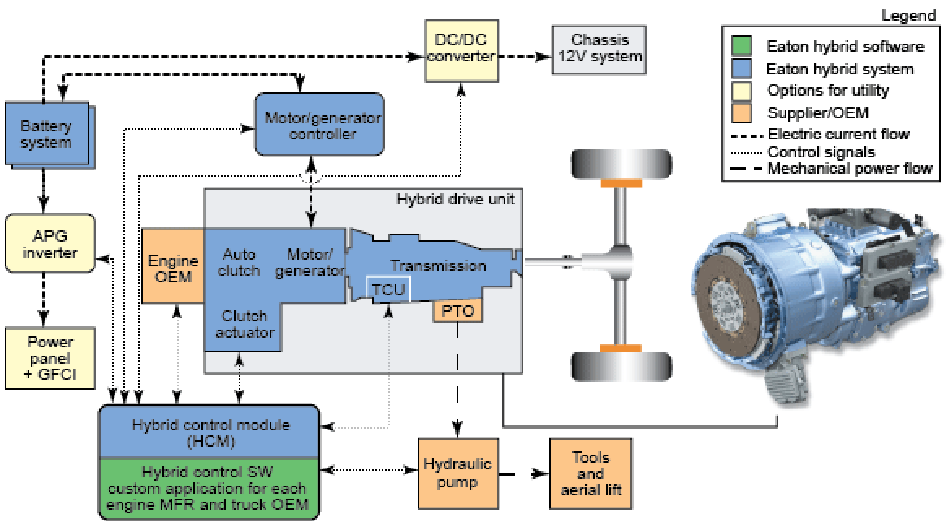
Identifying the need for variants

Designing for variants depends on the market maturity of the product and knowledge of how customers want to use the product. When a company introduces new technology, determining how customers will use it is challenging. Existing customers are not sure what changes they might need. With market uncertainty, the focus is on getting the product out the door to see how it performs, rather than on building a "product portfolio" or designing for variants.

When a product succeeds, the company gains market intelligence, and what variants are needed gradually becomes clear. For example, Craig Jacobs, Manager of Systems Engineering for the Eaton Corporation's Vehicle Group, says they created their first heavy-duty hybrid vehicle transmission variant, shown in Figure 1, by using the "clone-and-own" method. They made a copy of the first product and modified it. At that point, they had not identified the need for a full variant management process. A new design with some new components and software met the customer need. Then Eaton maintained both products separately.

However, as product improvements are made to one variant, they need to be reproduced in the second variant. This approach is not as efficient as true reuse, where modifications such as bug fixes that are made in one variant are automatically included in the second. As more variants become necessary, clone-and-own becomes less and less efficient.

Figure 1. The parts of the truck that Eaton builds



Base hybrid transmission system architectures from the same company might be the same around the world, but each customer requires the flexibility of interfacing with a different engine. These engines have electric motors with different power ratings and batteries with different capacities. Customers in different automotive markets also require the addition or subtraction of auxiliary electrical devices, such as AC power, heating, power steering, and so on.

Other industries have different but similar reasons to build variants. Developing variants requires that you identify the product behavior, divide that behavior into features, and vary the features. Variants can then be developed based on design rules. When you design a system with variants in mind, you anticipate where you expect variants to occur. This approach can be as simple as building clean interfaces or as complex as tagging the artifacts at potential places where variants might be needed (variant points).

Developing a variant management business strategy

A company goes through predictable steps in the transition from developing a single product to variant management. Typically, when requests for variants start coming in faster than a company can handle the requests, it is time to move to a design management strategy. Variant requests can come in the form of new requirements and behavior from customers and new suppliers from new markets. The transition from one design management strategy to another while supporting customers is a big decision. Management might not view such a big change positively.

To build a good business case, consider these factors:

Time to market and resource availability
If the opportunity window is small and your resource availability is limited, you might not want to design for variants at all. Instead, control might be easier if you can restrict the scope to a single variant.

Market understanding
Are you confident that the market will adopt your product? If not, do not invest the additional resources in variant design.

Global ambition
Nearly every country has unique rules and regulations. If you plan to compete in areas other than your home market, you will probably need a variant.

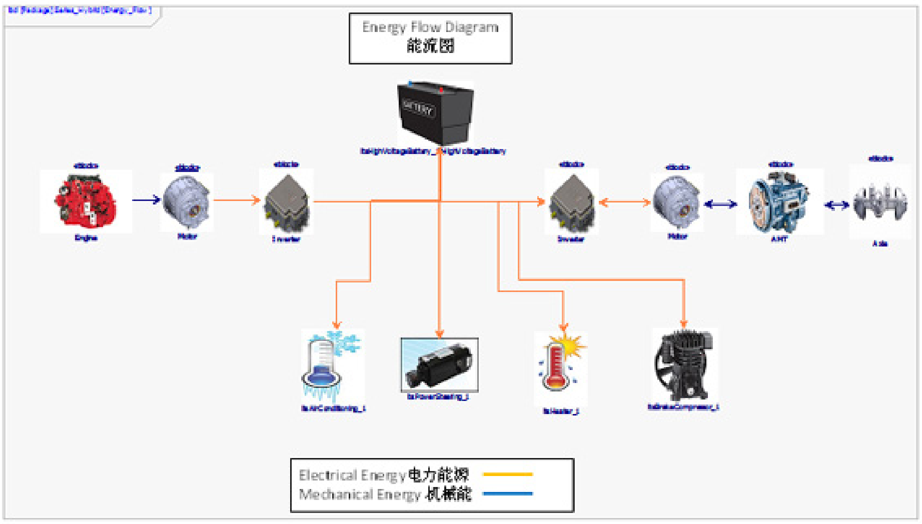
Product maturity
If you have not been developing your product for years, it is likely that you will need to vary and change your product as the product matures.

Customer behavior
Are you a subsystem supplier with multiple customers that demand differentiated final products? If so, designing for variants can help.

Gaining efficiencies through variant management

Companies shift to a variant management approach to improve their time to market and lower their maintenance costs. Using IBM® Rational® Rhapsody® modeling software in a systems engineering workflow to design for reuse and envision product line variants can provide efficiencies. Rhapsody helps you analyze and validate requirements, design rapidly with prototypes, and deliver more consistent applications by using the Systems Modeling Language (SysML) and the Unified Modeling Language (UML). This approach helps you design ease of integration of variants for OEMs and new component suppliers. For hybrid engine suppliers, for example, component interface specifications need to be written with the assumption that physical parameters will vary. Component and system software is designed for calibration capability to expand the product line efficiently. System functions are designed to be easily enabled or disabled.

Figure 2. Rhapsody diagram of the components of a specific variant



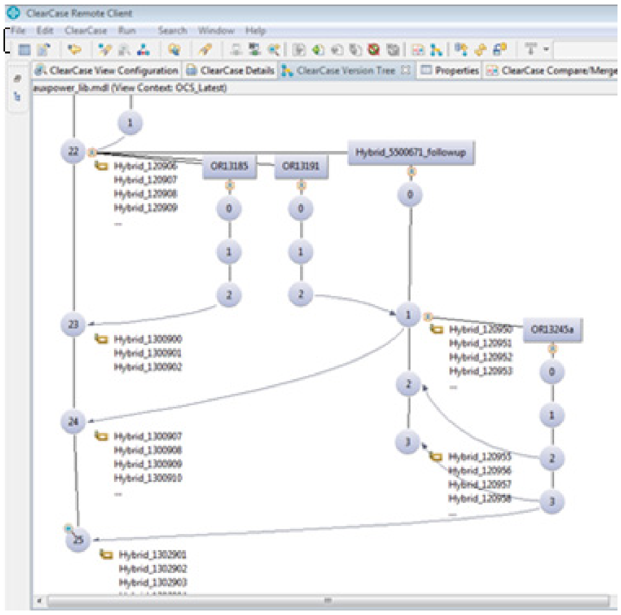
In Rhapsody, you can represent variants by using stereotypes and tags that are applied to a model element to indicate how the element changes for the variant. With this approach, you can explain the specific product variant more clearly. The stereotypes and tags can control the code generation of the final software that goes into the embedded device.

Engineers choose the level of abstraction to show the variant structures in Rhapsody. This design philosophy enables quick integration of new customer requirements and behavior into the system. Hybrid design groups can more efficiently support dozens of different customers, applications, and regional requirements.

A common solution in automotive companies is to load the appropriate variant into the code through a configuration file at run time. One main software file contains all of the necessary code, but the configuration file drives the decision about which code or set of calibrations to use for each variant. The reasons for using just one executable file are low volume, available memory, and ease of distribution to many locations across the company.

In combination with a load time strategy, you can use Rational® ClearCase®*configuration management software* to check in code variants as separate branches of a tree. With ClearCase, you can easily combine the branch variants. Automotive companies can merge regional requirements into global requirements and turn variants on and off, as shown in Figure 3. They create variants and then merge the variants into one deliverable. ClearCase supports views of different software code streams. By focusing on their own views, engineers can focus on their specific variants.

Figure 3. ClearCase view shows products diverging and then merged again



Deploying variants in software code

Variants in software code can be deployed in a product at several points in the development process:

1. **At code generation time.** Rational Rhapsody can generate code based on the variant that is selected for a specific product. The code is specific to the product and faster to read without code for other variants.
2. **At build time.** Use build time options such as these:
    a. Compiler directives, such as `#ifdef`
    b. Inheritance where one build includes a different child from another
3. **At run time (typically at initialization).** On initialization, the product configuration files are loaded into the application, causing it to use only designated variants.

Build time deployment is used when there is a high volume of code. However, build time variants can be hard to manage. For example, Eaton used this method previously, but they found that the overhead was too great, especially for managing many builds and ensuring that the right build was used for the right hardware variant. Significant overhead is associated with releasing software, releasing part numbers, and moving that process throughout the organization. With Eaton's low volume, lack of strong memory constraints, and need for fast time to market, they moved away from this method. However, medical device manufacturers and other industries are required to use build time deployment to satisfy regulations or processor memory constraints.

Avoiding common implementation problems

A typical challenge in developing for variants is overcoming the prevailing organizational and cultural mindset. In many projects, there is a clear need to convert to variant management, but there is "never enough time" given the current workload. Product managers agree that it is a good idea in theory, but they want the team to do the variant management work after their current project is finished. Product managers often feel that delays will be associated with converting to a variant development cycle, and they resist losing control over their individual projects.

When organizations finally decide to adopt a variant management strategy, there is a period of discomfort and program delays. The transition starts with uncertainty and resistance in the organization about whether conversion is the right decision. After the transition to a strong variant management process, they can enjoy faster time to market, higher-quality products, and more efficient engineers. Product line managers are happier because they have more predictable programs and resources. Customers are happier because they see higher quality and faster response times.

Craig Jacobs looks back in hindsight at his transmission project and wonders how they could have established a variant design methodology earlier in the development process. It might have taken them longer to get the first systems into the market, and they could have missed a window of opportunity. One compromise could have been to initially follow some design for variants best practices that require only minor work. That approach might have made it easier to build multiple variants later.

To identify a variant management business strategy, ask yourself these questions:

- Are there "easy" things that you can do in your initial design to support variants later if the product evolves?
- How can you create a flexible design up front without adding much cost?
- Can you make the design "too flexible" up front and add too much cost? This error could prevent you from realizing the benefit of product portfolios.

Best practices for product design for variants

These are key best practices for product design for variants:

- Design from the beginning with good, clean interfaces.
- Use good componentization.
- Create data variables rather than hardcoded constants.

When you follow these practices initially, it is much easier to adapt when the need for variants emerges. By following these practices, you can tag an element later and map the element to specific design features. Then, when you decide which development process to use, you will find it is easy to implement. Without a good design, this is a much tougher task.

You can use other IBM tools besides Rational Rhapsody and ClearCase to manage variants. Each tool supports different aspects of reuse and thus requires a different management approach. For example, Rational® DOORS® requirements management software supports versions to some degree through baselines and baseline sets, but it does not support variants of requirements or reuse of requirements. Therefore, to manage variants in DOORS, you need to use a clone-and-own approach. When you use IBM® Rational® Engineering Lifecycle Manager with DOORS, it can link the variants to each other and to the associated products.

Rational Engineering Lifecycle Manager also helps you manage variants. You can define your product lines and the variants in the product definitions, and link those product definitions to your requirements, designs, model elements, test cases, and other engineering artifacts.

Summary

No matter how you design for variants, they require more work than independent products, but designing for variants is often the most efficient practice in the long run. Determining when, why, and how to design for variants helps you manage your product portfolio so that you more tightly align your products with what your customers want over time.

Acknowledgement

The authors thank Craig Jacobs of Eaton Corporation for the contributions he made to this article.

Resources

Learn

- Find out more about the ways to use Rational software mentioned in this article:
  - Rational ClearCase for software configuration management
  - Rational DOORS for requirements management
  - Rational Engineering Lifecycle Manager to visualize, analyze, and organize engineering data and data relationships
  - Rational Rhapsody for model-driven development
- Explore the Rational software area on developerWorks for technical resources, best practices, and information about Rational collaborative and integrated solutions for software and systems delivery.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Attend a free developerWorks Live! briefing to get up-to-speed quickly on IBM products and tools, as well as IT industry trends.
- Watch developerWorks on-demand demos, ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.

Get products and technologies

- Download a free trial version of Rational software.
- Evaluate IBM software in the way that suits you best: Download it for a trial, try it online in a sandbox, or use it in a cloud environment.

Discuss

- Join the Rational software forums to ask questions and participate in discussions.
- Ask and answer questions and increase your expertise when you get involved in the Rational forums, cafés, and wikis.
- Join the Rational community to share your Rational software expertise and get connected with your peers.

About the authors

Joanne L. Scouler

Joanne Scouler is a curriculum architect at IBM, where she does business planning and course development for systems engineering software. She has developed and taught training courses on Rational Rhapsody software over the past six years. Her background in embedded systems and software modeling training has involved working with diverse clients, including Raytheon, Draper Lab, Pratt & Whitney, Zoll Medical, and Kollmorgren. Prior to IBM, Joanne worked at Telelogic, Hewlett-Packard, 3Com Corporation, Symantec, and Addison-Wesley. She holds a BA from McGill University.

Martin R. Bakal

Marty has more than a decade of experience working in various capacities in the embedded systems and software industry, with extensive customer experience worldwide in multiple industries, including consumer, telecomm, medical device, and automotive. He is currently the offering manager for the electronics industry at IBM Rational software. In that role, he leads the initiative to serve that industry. He has presented at many conferences and written magazine articles. Marty holds a BS in electrical engineering and a Master of Science degree in engineering management, both from Tufts University.