Product Line Requirements Management

Michael Sutherland
Galactic Solutions Group LLC
michael.sutherland@galactic-solutions.com

Abstract: Product development organizations use design reuse strategies to build common subsystems for the complex systems they develop and the many customers these systems are delivered to. Sub-systems are designed with an Architecture that will allow for strategic reuse. A base design for a sub-system is designed, with some reasonable modifications for specific Applications, to meet the requirements of the systems the subsystem will be incorporated into.

This paper provides details on a strategy for Requirement Management across Product Lines that has successfully been deployed within a large Product Development and Manufacturing organization.

Introduction: A Product Line is a group of products having a common set of features that are varied to satisfy the requirements of two or more Applications. A Product Line may initially consist of one Product designed for a single Application, which is then targeted for re-use with another Application. More mature product development organizations will engineer a common Product Architecture, which facilitates re-use and expansion of the Product Line. Domain expertise is leveraged to develop reusable core assets, which are then specialized by Application Engineers to develop the final products for specific Applications.

Example - Engine Product Line: In the automotive industry, a Vehicle incorporates a Powertrain to deliver and manage power to the Vehicle. Among its other components, a Powertrain uses an Engine to deliver power. A Vehicle is offered to the customer with more than one Engine option, such as a six cylinder Gamma V6 engine optimized for fuel economy and cost, or an eight cylinder Delta V8 engine optimized for power and performance.

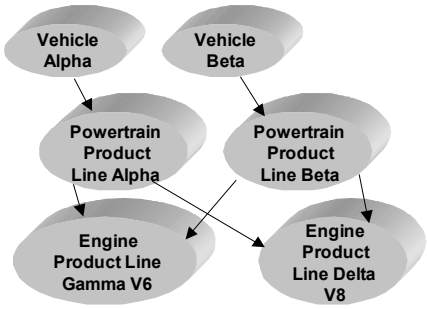

Figure 1 - Engine Applications

For the two Vehicles pictured in *Figure 1*, each is offered with two Engine options (Gamma V6 and Delta V8). These Engines are not completely redesigned for each Vehicle, but are specific Applications of Engine Product Lines.

Technical Specifications: For each Product Line, a Technical Specification is developed to manage the product requirements for the overall design of the Product, and the specific requirements for Applications of the Product.

A specific set of requirements for an Application is called a Configuration, and one will exist for each Application. Configurations are also used to specify requirements that are common across Applications.

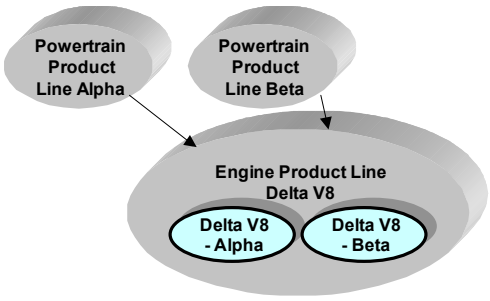

Figure 2 - Application Configurations

Example - Product Line Applications: *Figure 2* shows the two Product Application Configurations of the Delta V8 Engine that will be used in the Powertrain Alpha and Beta Product Lines.

TYPES OF REQUIREMENTS TRACKED

Product Commitment: Configurations marked "PC" are Requirement Values that are to be achieved for an Application Configuration.

Customer Requirement: Configurations marked "CR" are documented Customer Requirements, which an Application Configuration is designed to meet or exceed.

Design: Configurations marked "D" are common core Designs upon which specific Application Configurations are built.

Base: A Configuration marked "B" is designated to specify Requirements that are common to all Designs and Application Configurations.

CONFIGURATION RELATIONSHIPS

To aid in the reuse and eliminate re-specification of requirements within the Product Line, a hierarchical (tree) model defining the relationships between the Configurations is developed.

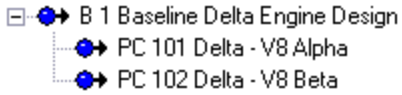

Figure 3 - Simple Configuration Definition Tree

Example - Configuration Definition Tree: The simple example in *Figure 3* shows that there are two Engine Product Applications. They are given a unique number for tracking (1, 101, 102, etc), and marked "PC". They are also represented as Child Configurations of a Baseline Design, marked "B". The relationship between the Configurations is represented in an Explorer-like view to represent the parent-child relationships.

Inheritance: A Child Configuration will inherit Requirement Values from the Parent Configuration if values are not specifically set for the Child. This process allows for the reuse of requirements for each Application without re-specification. Using the relationships defined in the Configuration Definition Tree, re-used Requirement Values need only be entered once.

C	R	Engine Technical Specification	Rel	Units	Base Delta	V8 Alpha	V8 Beta
		3.2.1.2.1 Engine Torque, Speed, and Power Ratings					
All	1	Engine Rated Peak Power	GE	HP	275	320	380
All	2	Engine Fuel Cut-Off Speed	LE	RPM	6000	[6000]	6200

Figure 4 - Requirements Specification

Example - Requirements Value Inheritance: *Figure 4* shows two Performance Requirements that have been specified for the Delta V8 Engine. The first has specific values set for the Baseline Design and each Product Application. The second shows that no specific value was set for the "V8 Alpha", so the value is inherited from the Baseline Design and is shown in brackets.

SPECIFICATION OF REQUIREMENTS

This compact line-item form for representing Requirements has been successful for the Engineers that are responsible for developing the Requirements. Often called an "expert view", it is the form desirable for day-to-day use of the Requirements by those who are intimate with them. Supporting and explanatory information is maintained in the same Technical Specification, and is "filtered" out to product the compact form. For verbose reporting purposes, the Requirement can be extracted programatically and written more naturally as a Shall statement.

Example - Natural Language Statement: In *Figure 4*, the first Requirement can be extracted as "Engine Rated Peak Power for the V8 Alpha Engine shall be greater than or equal to 320 horsepower."

Some Application Configurations will require the addition of new Requirements that do not apply to the complete set of Application Configurations. In this case the new Requirement is explicitly marked with the Configuration ID of the Application Configuration(s) it applies to. Inapplicability of a requirement is denoted by a special "N/A" marking in the Value field.

The concept of Requirement Value Inheritance can apply to other Attributes of the Requirements that are being managed, such as Rationale (reason or source for change), Validation Procedure, etc. Graphic Objects (charts, tables) can also be associated with Product Line Application Configurations when necessary, although they are not reported in Column form.

A primary benefit of tracking Product Line Requirements in the same Technical Specification is that a Column based side-by-side comparison for specific Values of each Requirement across the entire Product Line is a natural outcome. Columns can be selected and placed to meet the specific needs of a target audience and the Column layout can be saved for later retrieval. Also, access to edit a Product Line Application Configuration can be controlled per Configuration as necessary.

Other Configurations can be defined to manage Subject Matter Expert recommended values, and benchmark or trade-study values for similar products. These can also be viewed side-by-side in the context of the Product Technical Specification.

COMPLEX PRODUCT LINES

By defining a more detailed Configuration Definition Tree, many more Applications of a Product Line may be managed. New applications can be spawned off of any existing Application Configuration by adding new Child Configurations at the appropriate place in the Configuration Definition Tree. Fifty or more Configurations are not uncommon.

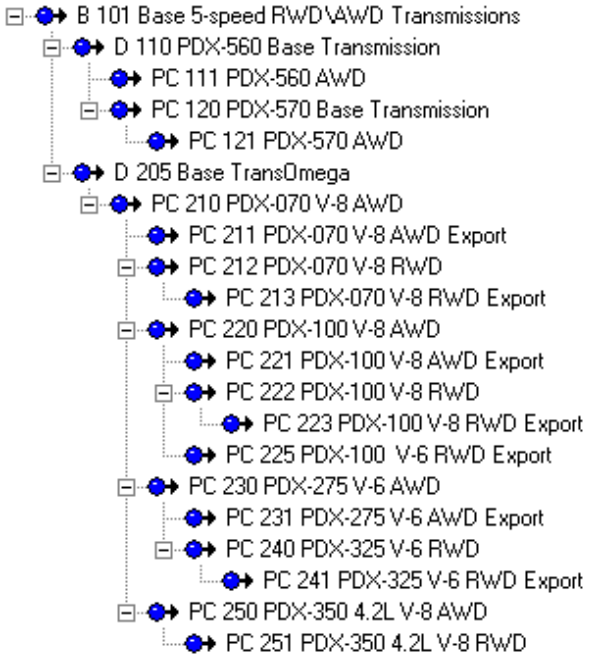

Figure 5 - Complex Configuration Definition Tree

Example - Complex Product Line Configuration Tree Definition: *Figure 5* shows a typical Transmission Application Configuration set, with variations for Export and Rear Wheel Drive (RWD) versions based off All Wheel Drive (AWD) versions.

USAGE

Every Product Line Application Configuration exists because it will be integrated into a larger system, or delivered to an external customer. These are also Product Lines, and as such will have their own Product Technical Specifications with Configuration Tree Definitions.

To comprehend Usage, traceability is established between Product Line Application Configurations and the higher level Configurations of the systems that they are integrated into.

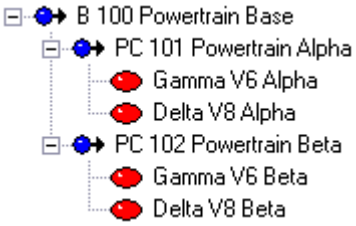

Figure 6 - Engine Usage for Powertrains

Example - Usage: *Figure 6* shows an example where Engine Usage is shown for Powertrain Application Configurations, corresponding to the example in *Figure 1*.

Building this knowledge into the process is essential for developing other kinds of Requirements Traceability through the Product Lines. Relationships between Requirements belonging to specific Application Configurations can then be formed to represent Functional Decomposition and Allocation, Interfaces, etc. Relationships can also be formed between Requirements from specific Application Configurations and Analysis Models, Test Plans and Procedures, etc.

OTHER CONSIDERATIONS

Configuration Management: Each Product Application has its own timetable for development and release. The entire Product Line is managed with a unified Configuration Management and Change Control Process. Changes to requirements are weighed carefully for their impact on the entire product line, either by an individual or by a Change Control Board, depending on the size and scope of the Product Line.

Change to Configuration Tree Definition: If the relationships between Configurations is changed, so will inheritance of Requirement Values between Configurations. The Relationships should be managed with this this in mind and the impact to affected parties determined so that no unwanted consequences occur.

Software Requirements vs. Hardware Requirements: The strategy outlined here is primarily being used to specify Performance Requirements for the hardware components of complex systems (engines, rudders, etc.) Software and Controls Engineers have a long history of successful re-use strategies, mainly focused on the application and configuration management of developed software components. Even though software components are increasingly designed to be re-used, more attention needs to be placed on re-use planning from the Requirements stage of development.

Software and Hardware Engineers also have very different styles of specifying Requirements. Software Engineers use Object-Oriented Analysis, Use Cases, and other methods to specify Requirements that may not be compatible with the methods used by Hardware Engineers.

Even though some of the key ideas behind the strategy outlined in this paper have come from work developed for Software Product Lines, it has yet to be seen if the strategies outline here will help bridge the gap between these differences.

Requirements Management Tools: As indicated earlier, side-by-side (grid view) reporting and editing of varying Requirement Values is a major benefit of this technique. A Commercial Off-the-Shelf (COTS) tool called DOORS, developed and distributed by Telelogic, was used for implementation of the technique. DOORS was chosen, among other reasons, for its ability to quickly display and edit information in a spreadsheet form (rows and columns) similar to the Microsoft Excel environment that Product Engineers were previously using to track requirements.

Implementation of the technique required the following enhancements to the core DOORS functionality:

- Definition of standard schema for storing Configuration Definition Tree, Requirements and Usage Relationships
- Script to read, interpret and display Configuration Definition Tree and Usage relationships.
- Script to allow user to select Configurations and Attributes to display (view builder).
- Script to update inherited Requirement Values, based on Configuration Definition Tree.

CONCLUSIONS

By using Product Line Requirements Management, development and production effort for common sub-systems is leveraged, centralized and minimized. The impact of change is clearly defined and can be assessed before changes are made to re-used Requirements. Usage is comprehended in the Design process, so that relationships between specific Application Configurations and Design elements that impact them can be formed and managed. New products can be added to the Product Line, or proposed products can be quickly analyzed for feasibility.

The strategy has proven to be comprehensive and extensible, and is the base for larger quality and process improvement efforts in the Systems engineering organizations that utilize it.

REFERENCES:

Bristow, David J., Bulat, Brian G. & Burton, D. Roger *Product-Line Process Development* Seventh Annual Software Technology Conference, http://source.asset.com/stars/loral/pubs/stc95/plpd95/sec0.htm, April 1995.

Fall, Robert H. & Steiglitz, Jennifer, *Comparative Evaluation of Product Line Requirements Architectures Developed at Honeywell* Proceedings of the Telelogic Americas' User Group Conference 2001, http://www.telelogic.com/usergroup/us2001/tracks_intermediate.cfm, July 2001.

Feiler, Peter H., *Configuration Management Models in Commercial Environments* CMU/SEI-91-TR-7 http://www.sei.cmu.edu/legacy/scm/abstracts/abscm_models_TR07_91.html, March 1991.

Jankun-Kelly, T.J. & Ma, Kwan-Liu, *A Spreadsheet Interface for Visualization Exploration,* Proceedings of IEEE Visualization 2000. http://graphics.cs.ucdavis.edu/research/Spreadsheets.shtml, October 2000.

Loureiro, G, Leaney P.G. and Hodgson M, *A Systems Engineering Environment For Integrated Automotive Powertrain Development,* Transactions of the SDPS http://www.sdpsnet.org/journals/vol3-4/loureiro.pdf, December 1999.

Software Engineering Institute -Carnegie Mellon University - *The Product Line Practice (PLP) Initiative,* http://www.sei.cmu.edu/plp/.

Vinga-Martins, Dr. Renato, *Requirements Traceability for Product-Lines* Workshop on Object technology for Product-line Architectures, http://www.esi.es/Projects/Reuse/Praise/pdf/ses2-3.pdf June 1999

Zak, Anatoly *Rockets R Us* IEEE Spectrum February 2002. http://caffeine.ieee.org/WEBONLY/publicfeature/feb02/rock.html

Biography: Michael Sutherland is the founder of Galactic Solutions Group, and has 12 years experience working with automotive suppliers and manufacturers. He has been a consultant to General Motors for 8 years, and is currently working with General Motors Powertrain and North American Operations (NAO) Manufacturing Engineering Divisions, developing and deploying Systems Engineering Processes and Tools. Michael has a Masters Degree in Electrical and Computer Engineering from Oakland University in Rochester MI. He also specializes in the Application of the DOORS Enterprise Requirements Suite, mentoring and teaching application, customization (DXL), and information modeling to a wide variety of clients across the nation.