PLM Neutral Customization Framework

By Shrikant Basarkod

Version 1.0
June 2009
Copyright Notice

© Geometric Limited. All rights reserved.

No part of this document (whether in hardcopy or electronic form) may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, to any third party without the written permission of Geometric Limited. Geometric Limited reserves the right to change the information contained in this document without prior notice.

The names or trademarks or registered trademarks used in this document are the sole property of the respective owners and are governed/protected by the relevant trademark and copyright laws.

This document is provided by Geometric Limited for informational purposes only, without representation or warranty of any kind, and Geometric Limited shall not be liable for errors or omissions with respect to the document. The information contained herein is provided on an “AS-IS” basis and to the maximum extent permitted by applicable law, Geometric Limited hereby disclaims all other warranties and conditions, either express, implied or statutory, including but not limited to, any (if any) implied warranties, duties or conditions of merchantability, of fitness for a particular purpose, of accuracy or completeness of responses, of results, of workmanlike effort, of lack of viruses, and of lack of negligence, all with regard to the document.

THERE IS NO WARRANTY OR CONDITION OF NON-INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO THE DOCUMENT. IN NO EVENT WILL GEOMETRIC LIMITED BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS, LOSS OF USE, LOSS OF DATA, OR ANY INCIDENTAL, CONSEQUENTIAL, DIRECT, INDIRECT, OR SPECIAL DAMAGES WHETHER UNDER CONTRACT, TORT, WARRANTY, OR OTHERWISE, ARISING IN ANY WAY OUT OF THIS DOCUMENT, WHETHER OR NOT SUCH PARTY HAD ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Confidentiality Notice

This document is disclosed only to the recipient pursuant to a confidentiality relationship under which the recipient has confidentiality obligations defined herein after. This document constitutes confidential information and contains proprietary information belonging to Geometric Limited, and the recipient, by its receipt of this document, acknowledges the same. The recipient shall use the confidential information only for the purpose defined above for which this document is supplied. The recipient must obtain Geometric Limited’s written consent before the recipient discloses any information on the contents or subject matter of this document or part thereof to any third party which may include an individual, firm or company or an employee or employees of such a firm or company. The recipient acknowledges its obligation to comply with the provisions of this confidentiality notice.
Contents

Abstract ............................................................................................................................... 4
PLM Customization Components ....................................................................................... 4
Pain Areas in PLM Customization ....................................................................................... 5
Solution Requirements ....................................................................................................... 6
Legacy Solutions .................................................................................................................. 7
New Classes of Solutions .................................................................................................... 7
Suggested Solution Approach ............................................................................................. 7
About the Author .............................................................................................................. 10
About Geometric ............................................................................................................... 10
Abstract
A Product Lifecycle Management (PLM) application is same for any organization’s product information as an ERP (Enterprise Resource Planning) application is to the organization’s human and transactional information. Proper selection of the PLM application and its effective implementation help the organization to manage multiple products efficiently. All PLM applications provide generic features mainly from product data management perspective. PLM vendors also provide Application Programmer Interfaces (API) and customization hooks to extend, configure or tweak the out-of-the-box (OOTB) functionality in order to fulfill the business objectives of the organization. However, these APIs and customization processes are proprietary to the PLM vendor. This white paper describes a few guidelines for customization, and typical issues faced during and after the customization phase. It mentions various tools available for overcoming the customization issues, including their limitations. Finally, the author presents an object oriented customization framework that is PLM neutral and provides significant benefits to the end users.

PLM Customization Components
Most PLM applications have at least a two-tier architecture, which means they have a server and many clients communicating with the server with different task requests. Some matured PLM applications like Teamcenter (Siemens PLM Software), PDMLink (PTC), and ENOVIA (Dassault Systèmes), also support four tier-architecture, wherein they have an additional web-tier (third-party application server) and resource (db) tier. Customization code can reside either on the client side or on the server side. Client side customization mainly includes UI related customization and user input validation. Change in OOTB behavior or extending the existing behavior is generally done on the server side. The business rules (in case of four-tier set up) are configured and customized in the application server environment. Typically, PLM customization can consist of following components –

- implementing predefined pre/post methods (server side)
- defining custom objects, new attributes and new methods (server side)
- implementing event based triggers, generally used in workflows (server side)
- processing objects in batch mode using standalone programs (client/server side)
- modifying the look and feel of the PLM application interface, adding new menus, options, dialogs, etc. (client side)
- interfacing or integrating external applications

It is generally recommended to customize not more than 20-25% of the process requirements or PLM features.
**Pain Areas in PLM Customization**

To be able to customize any PLM application one needs to be conversant with the following aspects:

- Programming skills
- API structure of the PLM application
- General PLM concepts
- PLM product specific knowledge
- Build and deployment process

Customization is virtually an ongoing phenomenon. During the initial implementation of the PLM application there would be more customization required. In the post implementation phase as the end users work more and more in PLM environment, new requirements start coming up, some of which need customization using APIs exposed by the PLM vendor. Customization tasks can be outsourced to a contractor or a system integrator. Irrespective of who faces the customization issues, at the end of the day it impacts the cost of the user organization.

**Following issues are quite commonly faced during PLM customization:**

1. **Resource ramp-up**: The customization engineer needs to have programming skills, knowledge of product specific APIs and its functionality, understanding of PLM concepts, and customization/deployment procedures. Hence, it is very difficult to induct a person into this role without intensive training, unless the resource has relevant experience in the past.

   Even after the training, it is advisable for the candidate to work on dummy or pilot projects initially, or bug-fixing assignments.

2. **Reusability of code**: In absence of a common framework for customization or codebase library, the code written by one person for a project is not leveraged or reused by another person for similar customization needs in another project. This results in significant duplication and redundancy in codebase, increasing the customization time and efforts.

3. **Code effectiveness**: PLM customization typically involves working on a sequence of APIs exposed by the PLM vendor. Same tasks can be accomplished using a single API or even multiple APIs. The use of inappropriate APIs often impacts the execution time and hence, the performance of the application as a whole.

   For different user requirements, the customization code written can have significant overlap. Hence, there are ample opportunities of code reusability, if it is written in an object oriented and modular fashion. Ad-hoc programming results in rewriting of the same code again and again. This also impacts the de-customization opportunities in future, when the subsequent product releases include customized functions in the OOTB solution itself.
Complex code leads to greater maintenance efforts post customization. Further, it cannot be used by others, due to its complexity reducing the code reusability. Many times combining business logic and presentation (UI) logic isolates the customization code from being reused and difficult to change and maintain.

4. **Deprecation of API:** With every new release of a product, the PLM vendor replaces many customization APIs with new APIs to improve performance or usability. Deprecated APIs, if used in existing customization, need to be replaced by the corresponding new APIs during upgrade or new release of the PLM application. Typically, this requires considerable efforts in modifying the code, followed by rigorous testing.

5. **Code review and testing:** Customization tasks, if not undertaken on a large scale, are generally developed and tested by the same person. Hence, there is no unbiased testing. Any logical or hidden errors results in undesired output or incorrect decision. For example, let’s assume that a certain workflow stage is auto-approved based on a business condition that is implemented through customization. If the code is not thoroughly tested for all use cases at run time, the workflow stage can get approved incorrectly, causing confusion for the concerned end users and stake holders.

6. **Function orientation in customization code:** Some of the major PLM vendors have imposed server side APIs in C. So the customization tends to be more of a function or task-oriented rather than object-oriented. Benefits of using object-oriented language for customization cannot be leveraged due to this limitation.

**Solution Requirements**

Organizations, more often than not, are reluctant to re-engineer their processes that are proven and have evolved over a period of time. They, therefore, have to resort to customization, if the process requirements are not met by the OOTB features of the adopted PLM application.

To counter the constraints and pain areas mentioned above, the user has to adopt a framework that eliminates these constraints, while improving productivity and quality. A solution framework that:

- facilitates reasonably quick ramp-up of general programmers into the PLM customization domain
- ensures code reusability and effectiveness
- is flexible enough to adapt to the changes in business logic
- is structured and scalable
- can interface and integrate with the upstream and downstream applications
- can adapt to changes in the PLM application
Legacy Solutions

There have been many attempts to address all or some of the solution requirements mentioned above; bigger the PLM implementation, more comprehensive are the initiatives in this direction. At the customer side a Utility Library with commonly used customization functions is developed. All customization engineers call functions from this library, thus reusing the code avoiding duplication and reducing the testing efforts.

A few System Integrators have looked at the problem more seriously from the productivity perspective. There are attempts to develop Code Generating applications that can take user inputs and generate the code in the background. This is more useful to define custom objects, new attributes on the existing objects, and to define new methods. It essentially provides a skeleton for the customization code. However, it is still function-based and needs frequent updates with any change in the customization procedure or APIs. Also, the reusability of code is not facilitated by this framework.

New Classes of Solutions

Similar to the code generating applications by the system integrators, recently the PLM vendors have come up with Business Modeler application, where the PLM configuration engineer can define business rules on different objects. Going further these definitions are stored in XML format, and can be very easily packaged and deployed on either Test server or Production server environment.

A more advanced approach is to integrate such business modeling applications with UML tools. PLM configuration engineer defines objects with its structure and inter-relationships; and this information is provided as input to the business modeler, which creates and defines the object structure, and the relations with given constraints.

Suggested Solution Approach

An application like Business Modeler by PLM vendors definitely reduces the customization tasks to a certain extent; however, the custom behavior is still largely accomplished through customized code. A Utility Library greatly facilitates reuse of frequently used functions; but it is still function-based and cannot leverage the obvious benefits of OOPS.

Going beyond the scope of customizing any single PLM application, the author here proposes and recommends a comprehensive solution framework for customization and collaboration purposes. This PLM-neutral customization framework is based on custom class hierarchy that is derived from the generic and common customization use case scenarios. Most of the PLM applications have almost similar features, in concept, from business point of view and differ only in its implementation. This conceptual behavior is captured in the custom class hierarchy at appropriate levels.
Custom classes include business objects, data objects as well as service provider objects.

Abstract classes constitute the core of the framework and include conceptual behavior of different PLM objects like an Item, Item Revision, Data Item, Document or Folder, and so on.

Concrete classes represent a specific PLM connector set and are derived from the abstract classes. These concrete classes implement PLM specific behavior only. So there can be multiple sets of concrete classes representing specific implementation behavior of different PLM applications, while the high level abstract classes are same for all these connectors, and provide the conceptual behavior.

Interfaces to these custom classes are exposed to the outside world (to the customization engineers’ community) so that the internal (conceptual as well as PLM specific) implementation is isolated completely from the customization code. Service provider classes contain the business logic, and use these interfaces to provide frequently used customization tasks like those in a typical Utility Library.

Benefits

Most of the problems faced by customization engineers today are addressed by the PLM-neutral customization framework. There is some maintenance activity involved for the PLM-specific connector part. However, this is required only when there is a new major release of the PLM
application. Following benefits clearly justify the efforts in developing and maintaining this type of solution framework:

- High reusability of the customization code
- Structured and scalable code, that can accommodate changes
- Significant reduction in defects and testing time
- Improved productivity
- Accelerated ramp up of customization resources
- Standardization in customization, easy to maintain
- Efficient collaboration with external applications
- Highly predictable quality of the code

**Limitations**

While the framework has plenty of benefits to offer, it comes with some implicit constraints and limitations. One has to seriously analyze applicability of these constraints to the business as well as PLM environment before proceeding further with this framework.

- The PLM-neutral customization framework essentially forms a wrapper over PLM APIs; it cannot be faster than PLM APIs if used alone. Performance does get affected for huge customization code. However, in the author’s view this limitation gets compensated by the reusability of efficient code written only once, and the productivity improvement accomplished through rapid development.

- The PLM specific APIs still need to be replaced in the concrete classes, whenever they get deprecated. However, this task is shifted from the customization team members to the support team of this solution framework. Also, the replacement can be done much faster due to optimum use of PLM APIs instead of at multiple places by individuals.

- It is possible that some customization tasks cannot be accommodated by the PLM-neutral customization framework, and hence the customization team member may have to resort to direct use of relevant PLM API. This practice, in the author’s view, is risky and may lead to confusion in future. One has to find a work around for such situation.

- The solution framework attempts to encompass the conceptual behavior from all major PLM applications. Hence, it is quite possible that for a particular PLM-specific implementation the customization engineer comes across extra parameters as input requirement or output result. This may cause some confusion, if the resource is not trained on the solution framework. Such excesses can be conveniently ignored unless mandatory.

At Geometric we have implemented a PLM neutral framework for one of our leading auto OEM customers. This implementation has evolved over a period of 4 years and has yielded significant benefits to the customer not only in terms of standardization but also in reducing customization time for new tasks by almost 60% and thereby significant savings on the cost as well.
To summarize, a PLM neutral customization framework is certainly an asset for system integrators engaged in regular customizations and for organizations that have matured PLM implementation incorporating or expecting to incorporate considerable amount of customization to map their product development process.

**About the Author**

Shrikant Basarkod is a Solution Architect and heads the Siemens PLM Product Development Practice at Geometric Limited. Shrikant has over 16 years of professional experience in the area of new product development including seven and half years in the domain of PLM applications and eight years in the automotive industry. He has worked on designing industry solutions on Siemens PLM Software’s Teamcenter platform for CPG and Retail industries, Enterprise Cost Management, APQP and Integrated Quality Management. The author has in depth knowledge of leading PLM applications as well as the industrial design and engineering processes.

**About Geometric**

Geometric is a specialist in the domain of engineering solutions, services and technologies. Its portfolio of Global Engineering services and Digital Technology solutions for Product Lifecycle Management (PLM) enables companies to formulate, implement, and execute global engineering and manufacturing strategies aimed at achieving greater efficiencies in the product realization lifecycle.

Headquartered in Mumbai, India, Geometric was incorporated in 1994 and is listed on the Bombay and National Stock Exchanges. The company recorded consolidated revenues of Rupees 5.98 billion (US Dollars 129.47 million) for the year ended March 2009. It employs close to 3000 people across 10 global delivery locations in the US, France, Romania, India, and China. Geometric is assessed at SEI CMMI Level 5 for its software services and ISO 9001:2000 certified for engineering operations. For further details, please visit [www.geometricglobal.com](http://www.geometricglobal.com).