Using a Software Product Line Approach in Designing Grid Services

R. Bashroush, R. Perrott

Belfast e-Science Centre, Queens University Belfast

Belfast BT7 1NN, UK

{r.bashroush, r.perrott}@qub.ac.uk

Abstract

Software Product Line engineering (SPL) has emerged in recent years as a planned approach for software reuse within families of related software products. In SPL, variability and commonality among different members of a family is studied and core assists (system architecture, software components, documentation, etc.) are designed accordingly to maximize reuse within the family members. In this work, we look at how this emerging technology can be relevant to the domain of grid computing and the design of grid services. The GeneGrid project is used to demonstrate the SPL approach.

1. Introduction

The concept of Software Product Lines (SPL) emerged from research in the areas of software reuse and domain-specific software architecture. Software product lines are a specialized form of software reuse as they employ planned reuse of software assets within the scope of a set of related products. As defined in [1], "*A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way*". A core asset is a software artefact that is used in the production of more than one product in a software product line. A core asset may be a software component, a process model, a plan, a document, or any other useful result of building a system.

There have been a number of formal processes defined for adopting an SPL approach within a software development environment. Examples are PuLSE™ [2], FAST [3] and RSEB [4]. These processes describe what the different stages of the SPL development process are. It is generally agreed that the first stage would be managing the variability within the product line and developing the product family environment (assets) and then using these common assets to develop product members as well as cater for family evolution and addition of new features/products to the product line.

In this work, an outline of the relevance of this emerging technology to the domain of grid computing and how it could be useful in designing grid services is given.

2. SPL and Grid

Software reuse continues to evolve from abstract data types, to objects (classes), components, and now services. Services introduce a larger level of granularity and reduced effort to integrate (reuse). They are characterized by their dynamic and loosely coupled nature compared to that of components. However, grid services exhibit some variability which could benefit from variability management techniques provided by SPL. Variability in grid services includes choosing the technology and platform, interface description, security policies, etc.

The SPL process can be tailored for grid service development and can be summarized in the following stages:

- *Variability management and feature modelling*: In this stage, the different features that are initially to be supported by the system are identified and the commonality and variability analysis among the different grid services (our product family) to be provided is studied and modelled. Existing variability management techniques can be used for this purpose [5, 6].

- *Asset development*: Once the system feature model is constructed, the second step is to develop the assets. The assets include the components constituting the services, the services, documentation, architecture, etc. Here, based on the nature of the *service family* (the set of services to be developed), two approaches can be used. A top-down approach where service descriptions are first

designed and then the constituting components. Or, a bottom-up approach, where first components are designed around the commonalities identified in stage one (variability management and feature modelling) and then composed together to form services. There are a number of techniques that can be used to help the architect with this stage including recently developed techniques [7, 8].

- *Evaluation and testing*: Once the services are designed, they are then evaluated against their pre-set quality (upgradeability, modifiability, etc.) and functional attributes. If satisfactory, they are implemented and unit level and service level testing are then conducted. There are a number of evaluation techniques which could be used in this stage [9, 10].

- *Evolution management*: Services introduced in the future (new versions, upgrades, etc.) can then be introduced making use of existing assets with minimal cost/effort (which is the main benefit of the SPL approach). However, to keep the *service evolution* (where the service code starts to deviate from the original architecture due to repeated modifications over time, and which could gradually render the service code unusable) minimal, changes should be introduced at the feature model and architecture level first, and then the modifications propagated down to affect the code (via component specifications, documentation, etc.).

3. Example

The example used in this section is a simplified version of the GeneGrid [11] project which was developed within the Belfast e-Science Centre (BeSC) in collaboration with Fusion Antibodies Ltd and Amtec Medical Ltd. The example is intended to demonstrate very briefly, due to space limitation, the general SPL approach for designing a grid based family of systems such as GeneGrid.

GeneGrid aims to provide a practical, easy to use and secure system, which harnesses and shares the power of distributed HPC resources, enabling more comprehensive and efficient interrogation of the global data pool available to biotechnologists. Additionally, the project aims to implement an underpinning scaleable and extendable architectural base, so that the addition of extra functionality, resources, or user capacity can be readily achieved.

Before starting with the feature model, we *scope* our product line by identifying what products lie within our product family. The products within the GeneGrid are the different UI portals (which invoke subset/or all of available grid services) that are designed to the specification of the end users. For example, the UI portal designed for Fusion Antibodies Ltd and Amtec Medical Ltd may differ from another portal developed for another company. The portals may differ in the number of services they access/provide (allowing for the development of low end and high end products). They could also differ in the way they provide the services (e.g. the way tasks and workflows are created, etc.). However, they all share the same core assets which are the GeneGrid services. Figure 1 shows an example of three products based on the same set of grid services.

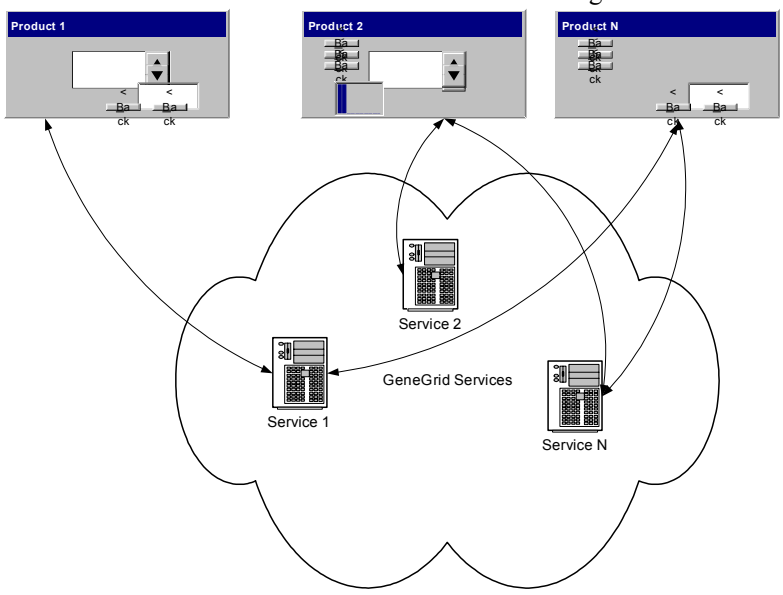

Figure 1. GeneGrid as a product family

After capturing the stockholders' requirements and specification, a feature model is built for the product family. Figure 2 below shows a small part of the GeneGrid feature model described using the FORM notation [6].

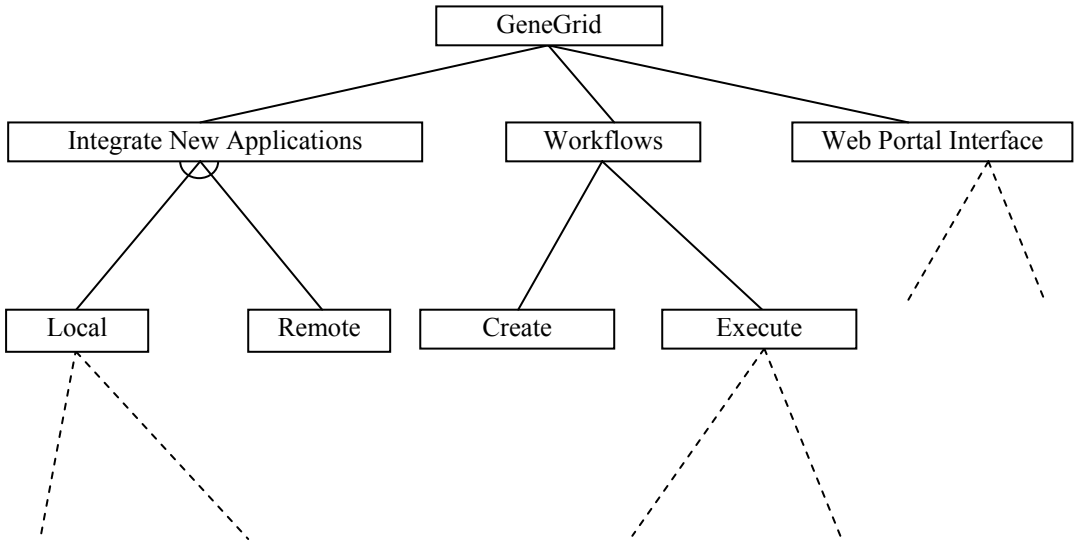

Figure 2. Part of the Feature Model of the GeneGrid system

The feature model above shows that the GeneGrid should allow for the integration of new applications locally or remotely. The applications that are available initially in GeneGrid are BLAST [12] (several variants), Transmembrane prediction and Signal peptide prediction. The set of application could vary between different products within the product line and also within the same product. The feature model also shows that the products should have a web portal UI and should allow the creation and execution of workflows. For more information on the FORM notation you can refer to [6].

Once the feature model is in place, the system architecture is designed. Figure 3 below shows the reference architecture of the GeneGrid system.

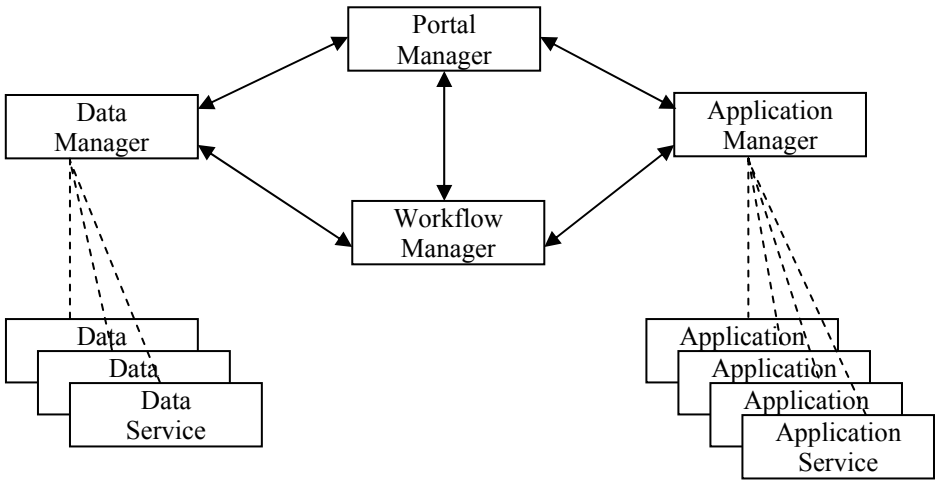

Figure 3. GeneGrid reference architecture

After creating the architecture, a number of scenarios are developed to test the architecture for its set requirements before investing any further in implementation. When found satisfactory, the components are then implemented and tested. For more information on the different components within the GeneGrid, please refer to [11].

Upon the completion of the core assets, the different products within the product family (Figure 1) are then constructed based on the desired feature set.

4. Discussion

The proposal is to use an SPL process for designing grid services. The process introduces concepts like *families of grid services* and *service evolution* and demonstrates how newly emerging technologies in software engineering can be used within grid computing.

Before using an SPL approach to develop a service family, a feasibility study is required. This is due to the fact that a substantial initial investment is needed to create the service family assets. During this time, no income/benefits can be expected from the activities carried out. Figure 4 below shows the economic model for adopting an SPL approach in order to develop a family of related products.

The GeneGrid project served as a good example for demonstrating the SPL process due to its nature where a fixed set of services are initially developed to be used by current and future products within the product family. Some components within GeneGrid, such as the GridManager component (refer to [11] for more details), were also successfully reused within other projects (horizontal reuse) at the centre.

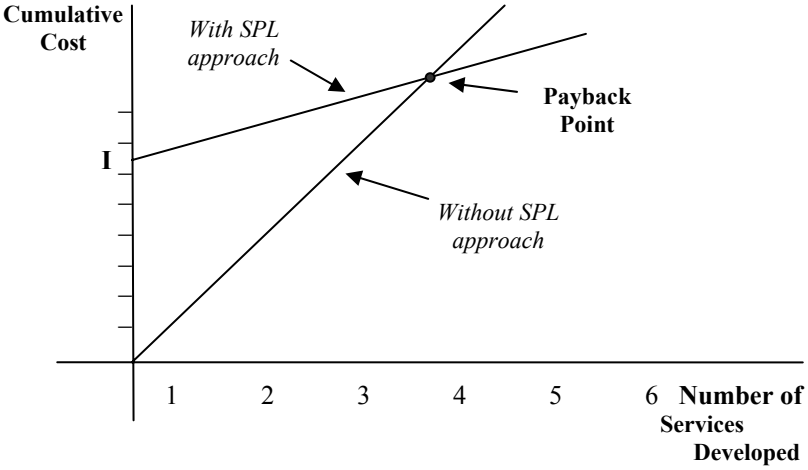

Figure 4. SPL Feasibility Analysis

5. References

[1] "A Framework for Software Product Line Practice - Version 4.2," http://www.sei.cmu.edu/productlines/framework.html.

[2] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J. D. Baud. PuLSE: A Methodology to develop Software Product Lines. *Proceedings of the Symposium on Software Reusability*, 1999.

[3] D. Weiss and C. Lai, *Software Product-Line Engineering: A Family-Based Software Development Process*. Reading, MA: Addison-Wesley, 1999.

[4] I. Jacobson, M. Griss, and P. Jonsson, *Software Reuse - Architecture, Process and Organization for Business Success*. New York: ACM Press, 1997.

[5] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Patterson, "Feature Oriented Domain Analysis (FODA) feasibility study," Software Engineering Institute, Carnegie Mellon University CMU/SEI-90-TR-21, 1990.

[6] K. C. Kang, S. Kim, J. Lee, and K. Kim, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures," *Annals of Software Engineering*, vol. 5, pp. 143-168, 1998.

[7] R. Bashroush, T. J. Brown, I. Spence, and P. Kilpatrick. ADLARS: An Architecture Description Language for Software Product Lines. *Proceedings of 29th Annual IEEE/NASA Software Engineering Workshop*, Greenbelt, Maryland, USA, April 2005.

[8] R. Bashroush, I. Spence, P. Kilpatrick, and T. J. Brown, "Deriving Product Architectures from an ADLARS Described Reference Architecture using Leopard," *ACM SIGSOFT Foundations of Software Engineering FSE-12*, October 2004.

[9] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for architecture evaluation," CMU/SEI-2000-TR-004, 2000.

[10] R. Bashroush, I. Spence, P. Kilpatrick, and T. J. Brown. Towards an Automated Evaluation Process for Software Architectures. *Proceedings of the IASTED international conference on Software Engineering SE 2004*, Innsbruck, Austria, February 2004.

[11] David R. Simpson, N Kelly, P.V. Jithesh, P. Donachy, T. J. Harmer, R.H. Perrott, Jim Johnston, Paul Kerr, Mark McCurley, Shane McKee. GeneGrid: A Practical Workflow Implementation for a Grid Based Virtual Bioinformatics Laboratory. *Proceedings of UK e-Science All Hands Meeting 2004 (AHM04)*, September 2004.

[12] Jithesh P. V., Kelly N., Simpson D., Donachy P. et al.. Bioinformatics Application Integration and Management in GeneGrid: Experiments and Experiences. *Proceedings of the UK eScience All Hands Meeting (AHM04)* September 2004.