A Software Product Line Engineering Approach to Building A Modeling and Simulation as a Service (M&SaaS) Application Store

Thesis

Presented in Partial Fulfillment of the Requirements for

The Degree Master of Science in the
Graduate School of the Ohio State University

By

Piyush Diwan, M.Tech.

Computer Science and Engineering

The Ohio State University

2013

Thesis Committee:

Dr. Rajiv Ramnath, Advisor

Dr. Jay Ramanathan

Dr. Thomas Bitterman
ABSTRACT

Modeling and Simulation (M&S) based design and manufacturing enables scientific and industrial communities to deliver results with greater agility and precision, and lower cost as compared to traditional methods. These benefits lead to increasing acceptance of M&S and emergence of advanced M&S applications. Despite recent advances in user interfaces, particularly web based, many M&S applications are still accessible only through command line interfaces on supercomputers that provide necessary computing and networking resources. Web-based interfaces could provide better accessibility, user experience and productivity due to user familiarity, enhanced visualization options, and wider application availability. These M&S applications often share similar architectural requirements besides the specifics. Considering such commonalities, following Software Product Line Engineering (SPLE) approaches would be valuable in cost-effectively building M&S applications.
Customized M&S applications deployed upon supercomputing infrastructure may enable supercomputing centers to holistically offer Modeling and Simulation as a Service (M&SaaS). The Ohio Supercomputer Center (OSC) has built an application store named **Ondemand.osc.edu** that packages a suite of web applications (portals) for a variety of M&S projects. In this thesis, we present an SPLE based approach to building application stores similar to OnDemand to support HPC based M&S. We explain how the common framework components and architecture of OnDemand were derived through and benefitted from application of SPLE methodologies. We propose an automation framework for building applications to provide HPC based M&S. We finally present a detailed analysis of our approach and challenges in adopting it.
Dedicated to my family.
ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Rajiv Ramnath, Professor Jayshree Ramanathan and CETI at The Ohio State University for their support and guidance throughout development of this thesis and my graduate studies as a whole.

I would also like to thank the entire Cyber-Infrastructure team at The Ohio Supercomputer Center including Dr. David Hudak, Dr. Thomas Bitterman, Eric Franz, Patricia Carey, and Shaun Brady for their significant part in influencing this work.

Lastly, I would like to thank my family for their unwavering support not only in my graduate studies but in all of my endeavors.
VITA

2004……………………………B. E., Computer Science & Engineering,
Government College of Engineering, Raipur, India

July 2005-June 2007………………M.Tech., Indian Institute of Technology Roorkee,
India

July 2007- September 2009…………Software Engineer, Symantec Corporation, Pune,
India

November 2009- January 2011……Senior Software Engineer, I-Flapp Technologies,
Bangalore, India

July 2011- present…………………Graduate Student, The Ohio State University

Field of Study

Major Field: Computer Science & Engineering
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................. ii

ACKNOWLEDGEMENTS ............................................................................................. v

VITA ............................................................................................................................... vi

TABLE OF CONTENTS ............................................................................................... vii

LIST OF FIGURES ..................................................................................................... xi

LIST OF TABLES ......................................................................................................... xiii

CHAPTER 1: INTRODUCTION .................................................................................... 1

1.1 Organization of Thesis ....................................................................................... 1

1.2 Problem Statement ............................................................................................ 2

1.3 Background ...................................................................................................... 2

1.4 Motivation ........................................................................................................ 8

CHAPTER 2: RELATED RESEARCH ...................................................................... 10

2.1 Related Research ............................................................................................ 10
2.1.1 HPC based modeling and simulation and related gateways ...................... 10

2.1.2 Software Product Line Engineering (SPLE) .................................. 12

CHAPTER 3: SOLUTION DESIGN .................................................................. 15

3.1 The OnDemand Application Store ....................................................... 15

3.1.1 OnDemand’s Architecture .............................................................. 17

3.2 Application of Software Product Line Engineering Approach ............. 22

3.2.1 Feature Modeling ........................................................................... 25

3.2.2 OnDemand’s central framework development ................................ 28

3.3 Generic SPLE process to build M&S application store ..................... 32

CHAPTER 4: RESULTS AND ANALYSIS ................................................... 43

4.1 User interfaces of OnDemand and its portals ..................................... 44

4.2 Validation of SPLE approach in building OnDemand ......................... 49

4.2.1 Quantitative analysis ..................................................................... 49

4.2.2 Qualitative analysis ....................................................................... 52

4.3 Problems/challenges faced in our approach ..................................... 54

CHAPTER 5: FUTURE WORK AND CONCLUSION ................................. 56

5.1 Future work ....................................................................................... 56

5.1.1 App Kit and App Runtime ............................................................... 56
5.1.2 Automation framework for portal generation ...................................................... 57

5.2 Conclusion .............................................................................................................. 59

REFERENCES .............................................................................................................. 61

APPENDIX A: SCREENSHOTS .................................................................................... 68

A.1. OnDemand Interfaces ......................................................................................... 68

A.1.1 Login Page ........................................................................................................ 68

A.1.2 The Dashboard .................................................................................................. 69

A.1.3 The Ajaxplorer File Browser .............................................................................. 70

A.1.4 The Cluster Terminal ........................................................................................ 71

A.1.5 The Job Constructor .......................................................................................... 72

A.1.6 The Job Monitor ................................................................................................ 73

A.1.7 Apps Menu ........................................................................................................ 74

A.2 FAN Portal Interfaces ......................................................................................... 75

A.2.1 Project Viewer .................................................................................................... 75

A.2.2 New FAN Simulation constructor ..................................................................... 76

A.3 Weld Predictor Portal interfaces .......................................................................... 77

A.3.1 Primary interface ............................................................................................... 77

A.3.2 New Weld Simulation constructor ..................................................................... 78
LIST OF FIGURES

Figure 1: Modeling advancements over time.......................................................... 3
Figure 2: Cloud organization of services, providers and interfaces........................... 6
Figure 3: OnDemand Architectural Overview....................................................... 18
Figure 4: High level representation of OnDemand’s framework stack ....................... 19
Figure 5: SPLE process to build OnDemand......................................................... 24
Figure 6: Functional feature model of Container Fill Portal...................................... 28
Figure 7: Generic SPLE process for M&S Application store development .................. 32
Figure 8: App Kit components.................................................................................. 35
Figure 9: The OnDemand Dashboard....................................................................... 44
Figure 10: Application Utilization by Category....................................................... 45
Figure 11: Primary interface of CFP......................................................................... 46
Figure 12: New job constructor interface of CFP...................................................... 47
Figure 13: Primary interface of SAWELD portal...................................................... 48
Figure 14: New job constructor interface of SAWELD portal................................. 48
Figure 15: Automation suite for building web portals.................................................. 58
Figure 16: OnDemand’s Login Page ............................................................................. 68
Figure 17: OnDemand’s DashBoard............................................................................. 69
Figure 18: The Ajaxplorer file browser ......................................................................... 70
Figure 19: Cluster terminal.......................................................................................... 71
Figure 20: The Job constructor ................................................................................... 72
Figure 21: The Job monitor.......................................................................................... 73
Figure 22: Apps Menu................................................................................................. 74
Figure 23: FAN Portal’s Project Viewer ........................................................................ 75
Figure 24: FAN Portal’s New simulation constructor................................................... 76
Figure 25: Weld Predictor Portal’s primary interface.................................................... 77
Figure 26: Weld Predictor Portal’s new simulation constructor interface..................... 78
Figure 27: Trucck Add-on Predictor Portal’s primary interface.................................... 79
LIST OF TABLES

Table 1: Brief description of some of the OnDemand’s portals .......................... 16
Table 2: Problem and solution domain specifications of product variants ................. 26
Table 3: Time and effort analysis of development process of different portals .......... 51
CHAPTER 1: INTRODUCTION

1.1 Organization of Thesis

Chapter 1 discusses the background of high performance computing based modeling and simulation and related software. It also formalizes the problem statement and motivation behind the solution.

Chapter 2 surveys existing research in High Performance Computing (HPC) based Modeling and Simulation (M&S) and Software Product Line Engineering (SPLE).

Chapter 3 explains the architecture and development process of OnDemand through SPLE approach. It begins by explaining the Feature Modeling process to identify common features among applications despite their heterogeneous problem domains. Then building of common reusable framework is explained.

Chapter 4 presents a detailed discussion and analysis of the SPLE approach for building OnDemand. It demonstrates the effectiveness of SPLE through quantitative and qualitative analysis and discusses technical challenges faced in this process.

Chapter 5 presents the anticipated future work and conclusion of this thesis.
1.2 Problem Statement

Industrial and scientific modeling and simulation carried out through web-based applications provides better user experience when directly using HPC services. Supercomputer centers can capture this opportunity and integrate their computing and networking resources with highly available web interfaces, providing a secure run time environment and isolated user workspaces. Building a foundational framework that caters to all such requirements and enables agile development of M&S solutions through plug-in components was the primary problem to be addressed. SPLE’s importance in developing such frameworks and comprehensive application stores also remains to be analyzed. In this thesis, we attempted to address these problems by explaining the OnDemand architecture and its correspondence with SPLE approaches.

1.3 Background

Manufacturing business communities have embraced modeling and simulation driven design processes at various scales in recent times. M&S based manufacturing is taking over physical product prototyping due to less expensive computer simulations, reducing time to market, while improving quality and cutting costs [14]. Industries have envisioned the worth of M&S and moved from physical modeling to coarse and high definition computational simulation based design. Figure 1 shows how manufacturing has advanced through several phases of M&S capability over the years. Without
computational tools, physical modeling via prototypes was and remains the only type of modeling available. Physical modeling tends to be an inherently slow, resource-intensive and error-prone process, yet is still often the only tool available to small and medium scale businesses.

![Figure 1: Modeling advancements over time](image)

Large organizations are increasingly adopting computational M&S to solve highly complex design and manufacturing problems, resulting in proliferating demand of M&S applications. They need highly scalable computing and networking resources to solve new problems swiftly and devise better solutions to old problems. Utilizing HPC resources to run industrial and scientific simulations has manifold benefits such as greater throughput, accuracy and scalability; a sped up design process, reduced research and development cost, and improved Quality of Service (QoS). The National Center for
Manufacturing Science (NCMS) reports that access to these technologies can reduce product design cycles by as much as 66% and lab testing costs by up to 98% [32]. A number of studies indicate that HPC based M&S is critical to the competitiveness of industries in the U.S. [39]. Therefore we need ingenious software tools to achieve high quality HPC based M&S.

Considering the critical role of HPC in M&S, provisioning approaches of HPC resources significantly influence quality of modeling and simulation outcomes. Web-based access to computing resources has become the default mechanism for many application areas, while HPC services continue to be accessed primarily through traditional interfaces as Secure Shell (SSH) (command line) and Virtual Network Computing (VNC) (visualization). A small number of science gateways [13] provide web-based access to HPC services though. These traditional mechanisms steepen the learning curve for new HPC users as they now have to learn a new interface paradigm in addition to the application itself [13]. Web interfaces could also improve currently-existing applications by providing a more natural interface for simulation setup (e.g., by allowing the user to visually manipulate the system to be simulated), thus helping lower barriers for new users to enter HPC based M&S by simplifying simulation-driven design process [13].
Gateways have not truly flourished in HPC due to development costs and administrative overheads required for each individual gateway [13]. It requires not only strong domain knowledge to build simulation software but also competence with web technologies to integrate them with cloud-based workflow and HPC resources accessibility. An appropriate blend of above expertise is needed to resolve development overheads in order to offer web-based solutions.

Supercomputing centers have exhibited limited engagement in developing domain specific simulation systems and primarily focused on providing HPC infrastructure and resources. However, San Diego Supercomputer Center (SDSC) [39] and Ohio Supercomputer Center [34] among others have collaborated with domain experts to assimilate their expertise and have built customized portals for scientific simulation problems to provide user-centric access to their HPC services. This indicates a growing trend of adopting web-based HPC. Figure 2 shows the cloud organization of services, their providers and the interfaces, in particular web-based [14].
The integration of HPC with web-tools to solve M&S problems is known as Modeling and Simulation as a Service (M&SaaS). Providing M&SaaS through the web has manifold benefits such as (a) reduction in cost and overhead associated with hardware and software deployment, upgrade, maintenance and other associated costs [13, 14] as users need to bear the cost associated with only actual usage cycles (pay-as-you-go), (b) improvement of user experience through fast and attainable web access to HPC services. In general, M&SaaS benefits the technology and infrastructure providers, scientific communities, large-, medium- and small-scale businesses at various levels and scales.
Current usage patterns of M&S applications indicate that the majority of them are subjected to access HPC resources in a secure environment, despite the differences in simulation software they run at background and their input and output formats [13, 14]. Therefore their underlying architectures ought to have common components that serve common functionalities. Given these characteristics, shared frameworks need to be built that can be reused to build individual software and facilitate agile development.

Software Production Line Engineering (SPLE) defines methodologies to build set of software that satisfies specific needs of consumer segments through sharing a common, managed set of features developed from a common set of core components [6, 8, 9]. SPLE avoids the need to develop a new product from scratch for each customer, encourages software reuse, and enables rapid market entry [8]. Therefore SPLE can be constructively used by M&S service providers to cost-effectively meet user demands building suites of applications.

Feature modeling is the first step in building a series of product variants with significant commonalities and variations. Feature models are simple, hierarchical models that capture the commonalities and variability of a product line [6]. In feature modeling, the product line's commonalities and variability are described in the problem space, while the solutions are typically results of SPLE. This reflects the desired range of applications in
1.4 Motivation

The Ohio Supercomputer Center (OSC) has been a primary source of High Performance Computing services providing high-end computing, storage and networking resources to many industrial and academic research projects [13, 14, 34]. Much of OSC’s load is composed of computational and networking oriented simulations for large-scale problems [13]. Typical simulations are run by creating job scripts and submitting them to a cluster’s batch processing system followed by visualization after completion of the job [13, 14]. Here isolation of the user’s data and application execution are of primary concern. Advanced users may work with terminal and VNC interfaces but the majority of other users increasingly need web-based solutions that provide ways for submitting, running and tracking computational jobs, scheduling them on compute clusters, generating reports, and allowing direct access to user workspace on clustered HPC file systems [13, 34]. OSC aims to support both kinds of users, those who work with web tools and those who apply traditional tools such as ssh, sftp and VNC [13].

A unified web platform might be essentially useful to cater to a variety of user’s needs where users could access science gateway-style web applications for M&S, VNC
applications, and HPC centric file systems [13]. Web infrastructure research at OSC had earlier provided high quality solutions to address these needs but in a segregated form. An integrated application store as a single point of entry to OSC’s HPC environment would provide wider usability to users. This desirability motivated development of the OnDemand gateway to offers features mentioned earlier.

The cardinal features of OnDemand that reflect its core capabilities and shared among individual portals can be standardized as secure user workspace, isolation of job and user data, job management and so on. OnDemand’s architecture was built upon such foundational components as App Runtime [13], a runtime environment component for secure access, and Per User Drupal, a Drupal [16] based, easy to deploy framework for developing product line of web portals. Standardization of features has motivated us to relate them to SPLE methodologies and analyze the impact of reusable frameworks in developing pluggable customized components around this framework and subsequently building M&S application stores such as OnDemand. We will discuss details of the framework, its components and their development process in subsequent chapters.
CHAPTER 2: RELATED RESEARCH

2.1 Related Research

In this chapter we will go through a variety of software applications that function on modeling and simulation problems. We will first discuss science gateways (portals) that operate in HPC environments to achieve higher throughput and efficiency. Subsequently we will discuss work in the area of Software Product Line Engineering to understand how it has helped in developing homogeneous software applications.

2.1.1 HPC based modeling and simulation and related gateways

With the high adaptation rate of M&S based design, science gateways and grid services have emerged through support of significant work done in this area. Globus Toolkit [1, 18, 19] an open source toolkit for grid computing developed and provided by the Globus Alliance, provides fundamental technologies for identification, authentication, authorization and service discovery. For example, the Computational Chemistry Grid project created and maintains the GridChem science gateway [15], using Globus
technologies. GridChem users from multiple institutions can securely access services from multiple resource providers [13, 15]. GridChem is also illustrative of use of science gateways to form *virtual organizations*, where participants from multiple physical institutions can perform calculations and share data via a common web platform, thus forming a single virtual organization [13]. The San Diego Supercomputer Center (SDSC)’s Trestles project focuses on establishing a portal on the TeraGrid for structural biology researchers to facilitate electron microscopy (EM) image processing using the Appion pipeline, an integrated, database-driven system [39].

The eXtreme Science and Engineering Discovery Environment (XSEDE) User Portal (XUP)(portal.xsede.org) provides integrated GSI-SSH, job status, machine status and account management [48]. OnDemand includes similar functions in addition to file browsing and editing, job control (job submission and deletion) and VNC application access [13]. The nanoHUB project [26] has released the HUBzero [30] software distribution for creation of virtual organizations at institutional (or even laboratory) level. The Open Science Grid also supports non-browser applications for submitting HPC jobs from desktops, like BoSCO [5]. UNICORE [43] supports web portals through its Portal Task Force [44].
Current efforts address basic services, security, and workflow management. OSC’s web infrastructure team has also developed web tools in remote instrumentation [7] and data-intensive biomedical science [22] areas. These applications were built as standalone web applications that provide user isolation at the application level and are deployed in community accounts, e.g. a single HPC account holding users’ data [13, 14].

2.1.2 Software Product Line Engineering (SPL)

SPL [4, 6, 8, 9, 10, 11] is a commonly practiced approach in software engineering. Significant work has been done to construct similar web tools by effectively exploiting software reuse [3, 10]. Beuche and Dalgarno [4] demonstrated application of SPL to develop similar products in different problem contexts. Pettersson and Jarzabek [42] presented their experience in building a web portal product line using reactive approaches to SPL. Feature modeling [4, 25, 28] provides simple and hierarchical models that capture commonalities and variability of a product line. Hence it is used as a basis to design and architect many product lines [4, 14, 27]. Feature modeling mechanisms were still useful in defining feature variants and invariants among individual portals that are part of the OnDemand application store.

Software reuse techniques presented in earlier work were relevant while developing OnDemand. To the best of our knowledge, no work so far has utilized SPL approach to
build application store as single point of entry for particularly enabling HPC based M&S. Our problem contexts were significantly different than contexts presented in earlier work, particularly in underlying M&S problem specifications, security and privacy concerns, technologies used and accessibility requirements. However resultant benefits were similar. The SPLE approach could streamline development work of common modules and the process of integrating them into a reusable framework.

End User Computing (EUC) [29] is a modern trend of involving end-users in the software engineering process with different roles and capabilities. Software products have added tremendous values to their user experience and market reputation of those organizations practicing EUC. Industries have incorporated EUC at various levels to improve their user’s experiences through greater involvement. In chapter 5 we propose the architecture for an automation suite, to enable EUC in portal development process. This architecture can potentially expedite portal development process and optimize user experience.

The Rappture toolkit [38] supports Rapid application infrastructure development, making it quick and easy to develop powerful scientific applications. VMware has implemented EUC with its View, ThinApp, Horizon and Mirage [45, 46] applications to modernize desktop infrastructure of end-users. View and ThinApp allow users to manage traditional desktop layers (operating system, applications and persona), while
View and Mirage allow administrators to provision and set policies to new or groups of
desktops [45]. Novell Inc has facilitated EUC in **Collaboration** and **End-Point**
**Management** [33] solutions that helps users to simplify and reduce cost of endpoint
device lifecycle management across platforms and locations. Our approach differs from
existing solutions in elevating the user’s involvement in portal customization by virtue of
the proposed automation module in addition to OnDemand’s foundational framework that
will be described in the next chapter.
CHAPTER 3: SOLUTION DESIGN

In this chapter we explain our approach to building the OnDemand application store through the application of SPLE in order to build a common reusable framework. We begin by explaining the high level architecture of OnDemand. We then explain the feature modeling and SPLE processes involved in building common framework components and in turn various M&S applications. We finally extract a generic SPLE approach that can be used to build HPC oriented gateways based on our experience.

3.1 The OnDemand Application Store

The OnDemand is a comprehensive, dashboard-like, application store, designed and developed by The Ohio Supercomputer Center (OSC)’s cyber-infrastructure team. It provides an integrated, multi-purpose, single point entry to its HPC resources, file-system, and a variety of gateway-like M&S applications. It integrates transparent HPC usage interfaces such as a terminal, file browser and system usage monitor along with
science gateways that hide HPC usage behind a web interface’s workflow [13]. Table 3.1 provides a list of some of M&S applications that are available through OnDemand.

<table>
<thead>
<tr>
<th>Portal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heat Sink Predictor (HSP)</td>
<td>Simulates and predicts the performance of heat sink designs.</td>
</tr>
<tr>
<td>Manifold Flow Predictor (MFP)</td>
<td>Simulates Newtonian fluid flow through a manifold with one inlet and multiple outlets.</td>
</tr>
<tr>
<td>Container Fill Portal (CFP)</td>
<td>Simulates filling operations of fluids in containers.</td>
</tr>
<tr>
<td>Truck Add-on Predictor (ACP)</td>
<td>Simulates truck models.</td>
</tr>
<tr>
<td>Job Inspector Portal (Queue)</td>
<td>Status inspection of all jobs running on different clusters.</td>
</tr>
<tr>
<td>Job Constructor Portal (Job)</td>
<td>Job specification and submission on different clusters.</td>
</tr>
<tr>
<td>SA Weld Predictor (SAWELD)</td>
<td>Simulates underwater industrial weld processes.</td>
</tr>
<tr>
<td>E-Weld Predictor (EWELD)</td>
<td>Simulates industrial weld processes.</td>
</tr>
<tr>
<td>Horton Fan Portal (FAN)</td>
<td>Simulates fan models.</td>
</tr>
</tbody>
</table>

OSC’s computing and networking resources have been accessed through command line, standalone web applications and other user interfaces to run a number of scientific and industrial projects. A single HPC account holds users’ entire data across gateways, while isolation of user-workspaces is achieved at the application level only [13]. This leads to an overhead of managing gateway deployments and accounts on a per-gateway basis. OnDemand addresses these issues through multiple services reachable from a single entry point that relies on common authentication [23]. Thus users can reach gateway functionalities and at the same time have system-level access to their data (e.g. access at
the command line, file-system browser, VNC access) [13]. Figure 3 shows a dashboard as a navigational aid presented upon login as an example of an application running under this architecture. For more details and visual representation of OnDemand interfaces see Appendix A. In summary, the major salient features of OnDemand are [13]:

1. A unified point of access for all OSC services,
2. Single sign-on,
3. Zero install (no local SSH or VNC client installation or configuration required of the user),
4. Distributed files-system access (upload, download, browse and edit),
5. Create, control and monitor jobs,
6. Access shells on cluster login nodes or launch a set of web and VNC apps.

3.1.1 OnDemand’s Architecture

OnDemand’s architecture has been built around the OnDemand Runtime (ODR) environment. In other words, the core functionalities offered by OnDemand are controlled through ODR. ODR has five major components: Dashboard, Web Service, Per-User Web Server, Proxy, and OpenID Server. Figure 3 depicts the workflow among these components. Figure 4 represents the OnDemand’s framework stack at high level. A brief overview of the framework components is given subsequently. In addition to these
major components, there are other auxiliary modules as well that function cohesively to provide functionalities mentioned earlier.

Figure 3: OnDemand Architectural Overview
1. **OnDemand Dashboard**

OnDemand’s Dashboard is the upfront interface that is presented upon a user’s login. It is primarily responsible for authentication, authorization, and accounting (AAA) [13]. It extracts the common AAA processing from each gateway, allowing them to perform only their intended tasks, thereby avoiding redundant AAA operations. It authenticates users as web-user (by running as apache user) and an OS-user (through OpenID). This dual authenticated user can broadly be termed as ODR-user [13]. The communication with
gateways occurs only through the ODR Web Service to support multi-host, or multiple Dashboards, on single OnDemand Web Service environments [13].

2. OnDemand Web service

The ODR Web Service instantiates a web or VNC server as a specific web user upon request through the Dashboard. Then it starts the requested portal on that web server instance and returns a URL at which that portal can be accessed [13]. On logout request, the ODR Web Service stops access to the applications by shutting down the Per-User Web Server [13]. This service abstracts out these important common functionalities from the portals. If all the portals were to run as stand-alone applications then they would need to perform these operations individually. This would overload the server with per user per portal instance of the service. Hence providing this functionality through a common service is naturally a better approach.

3. Per-User Web Server

A Per User Web Server (PUWS) is the ODR-user specific instance of an Apache web-server that runs to ensure that every ODR-user’s client may connect to only processes owned by that ODR-user [13]. This is important because ODR-users also have OS credentials on the server, so it is necessary to prevent users from accidentally or intentionally accessing another user’s data. All portals run on these PUWS. In this
manner all HPC jobs, visualizations, and data transfers run as the ODR-user [13]. In the absence of PUWS, every portal would run with same web server credentials, which would be more difficult to protect, as a compromise to the common web server would provide access to all user’s metadata. PUWS may alternatively be termed as Per-User Apache (PUA). We developed a Drupal [16] based template that encapsulates PUA, called Per User Drupal (PUDL) and is reusable in building different portals [14].

4. OnDemand Proxy

The OnDemand Proxy redirects HTTP requests from authenticated clients to their corresponding PUWS process at the appropriate port number [13]. This port number is identified by the OpenID authentication token provided by the Runtime against the username [13]. Ultimately, it ensures that only the PUWS owner should be able to access the specific portal. It is important to note that such an authentication process would lead to great overhead on individual portals if this common functionality were not abstracted into a common framework.

5. OpenID and Single Sign-On (SSO)

The OnDemand single sign-on feature allows users to have single point entry to its various portals and other HPC interfaces. This feature has been implemented through OpenID [13] authentication. OpenID is an open standard that allows for co-operating
sites to use a third-party service for authentication [13, 35]. It has been deployed by many large websites including Google, Yahoo!, and IBM [17, 20, 49, 36]. The authentication mechanisms are transparent to the applications so as to allow other applications to function seamlessly that depend on SSO [13]. As a part of SPLE process, including SSO in the ODR framework eliminates the redundancy of the authentication process in individual portals.

3.2 Application of Software Product Line Engineering Approach

The Software Product Line Engineering approach influenced the development process of OnDemand, especially feature abstraction and modularity of architectural components. The aforementioned framework components eliminated the redundant implementation of corresponding features in each web application and provided the platform for developing individual applications under one umbrella: The OnDemand architecture. Additional benefits will be discussed and analyzed in chapter 4 in detail.

Application of SPLE approaches is a three step process. The first step is to identify features and requirements of the system including underlying product variants to be built. Then common and unique requirements are segregated and mapped to corresponding features or modules in the solution. This step is accomplished through Feature Modeling.
The next step is to build those modules or features, preferably as part of a framework that can be reused by multiple variants to address common requirements. The final step is to implement specific features that can be plugged together with framework components to build customized product variants. These steps are shown in Figure 5 and elaborated below as well in OnDemand’s context.
Figure 5: SPLC process to build OnDemand
3.2.1 Feature Modeling

The web applications that are developed at OSC share many common requirements, primarily concerned with system access and security mechanisms. These requirements along with the specific features define the problem space in feature modeling. We identified these common requirements and their inter-dependencies from the initial portals that were developed as stand-alone application. The solution space is described by constituent outcomes of the product line that implement the common features and their correspondence with the problem space. Examples of such outcomes in OnDemand’s context are the ODR components described in section 3.1.1, and Drupal [16] based template called PUDL [14] and its underlying modules.

A two-dimensional division of features resulting from feature modeling on OnDemand’s portal variants is shown in Table 2 in decreasing order of their priorities [14]. This table shows correspondence of features in problem and solution spaces with the distinction of their commonality and uniqueness. A notable point here is that unique features such as “visualizations tools/packages” are subject to customizations of common features such as “ability to run visualization tools for result analysis”.

25
### Table 2: Problem and solution domain specifications of product variants

<table>
<thead>
<tr>
<th>Feature requirement in Problem Domain</th>
<th>Software components in Solution Domain</th>
<th>Implementation type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure user workspace.</td>
<td>Per User Web Server.</td>
<td>Common framework</td>
</tr>
<tr>
<td>Common access point.</td>
<td>SSO and ODR DashBoard.</td>
<td>Common framework</td>
</tr>
<tr>
<td>Common authentication across portals.</td>
<td>ODR Proxy, SSO.</td>
<td>Common framework</td>
</tr>
<tr>
<td>Interface to provide parameters for job creation.</td>
<td>Web form layouts.</td>
<td>Template customization</td>
</tr>
<tr>
<td>Interfaces for view/edit/delete operations of existing jobs.</td>
<td>DB management components.</td>
<td>Template customization</td>
</tr>
<tr>
<td>Submit simulations job.</td>
<td>Job submission, execution, status inquiry scripts.</td>
<td>Template customization</td>
</tr>
<tr>
<td>Run visualization tools for result analysis.</td>
<td>Common modules for invoking 3rd party applications/tools.</td>
<td>Template customization</td>
</tr>
<tr>
<td>Tailor look and feel of the interface.</td>
<td>Common widgets, themes, menu items.</td>
<td>Template customization</td>
</tr>
<tr>
<td>Portal parameters based on the problem specifications.</td>
<td>Customizations of web-forms, widgets, themes etc. to reflect portal requirements.</td>
<td>Separate implementation</td>
</tr>
<tr>
<td>Simulation scripts, tools, programming languages etc.</td>
<td>Generic interface to utilize different simulation packages.</td>
<td>Separate implementation</td>
</tr>
<tr>
<td>Job management features e.g. export/import, duplications etc.</td>
<td>Customizations of scripts for individual portals for specific job-provisioning features.</td>
<td>Separate implementation</td>
</tr>
<tr>
<td>Visualization tools/components.</td>
<td>Integration with 3rd party visualization tools/packages.</td>
<td>Separate implementation</td>
</tr>
</tbody>
</table>
Figure 6 shows a functional feature model of one of OnDemand’s product-line portal named *Container Fill Portal* [14]. It is one of the **National Digital Engineering and Manufacturing Consortium (NDEMC)** [32] projects, and is used to run simulations on various container models with different kinds of fluids. The feature model is represented as a tree where the root represents the variant itself. The next level of nodes represented by gray boxes depicts the mandatory features. The white boxes denote the optional ways to address mandatory features, while the blue boxes represent only two possible options for a particular feature. Notice that the mandatory features correspond to some of the problem domain features listed in Table 2. Feature models for other portal variants may also be derived in a similar fashion that reflects the common and specific features in problem and solution domains. The results of feature modeling establish a list of common features to be implemented as framework components. Next subsection explains this process in more detail.
3.2.2 OnDemand’s central framework development

Feature modeling provided the common and unique features of portal variants. As the second step, the solution space components that address common features were implemented to form a common framework. This framework includes the core components of ODR and a Drupal [16] based template named Per User DrupaL (PUDL) [14] that aids in developing future portals under OnDemand’s architecture. ODR components were described in section 3.1. In this subsection we describe the PUDL template and its applications.

Per User DrupaL (PUDL) Template

Drupal [16] is an open source content management system composed primarily of PHP, CSS, and JavaScript. It provides agile ways to write web-form based interfaces. It is well
suited to develop the M&S web applications that are demanded by OSC users. However some of Drupal’s limitations such as centralized Apache configuration etc. required us to modify its functioning before it could be used in our environment.

Web applications traditionally manage user accounts at the application level while OSC’s users need access to HPC data and resources accessed via a UNIX user and group id [13, 14]. Furthermore, redundant installation of required applications such as SQL, Apache, and Drupal in each user’s workspace would greatly consume system space. Therefore the Per User Web Server (PUWS) model was adopted where a single installation of this software was instantiated by multiple UNIX user accounts in order to meet the above specified requirements. Drupal is used for developing web modules around the PUWS model. Some of the reusable modules from earlier portals were harvested together to form the PUDL template. It optimized development of future portals under the OnDemand application store in terms of time and effort. Some of these reusable modules are listed below:

1. Webforms (PHP, html, JavaScript).
2. Perl scripts for parameter replacement, job setup, job submission, and status inquiry.
3. Web interface themes.
4. Portal resource management.
5. Path qualifier code.
6. Modules for invoking 3rd party visualization tools on completed jobs (e.g. ParaView and Abaqus).

Benefits of PUDL template

- Reusability of PUDL modules in developing new portals saves time and effort significantly.
- User has complete control over any uploaded information (files and data). As these files are uploaded into their system account, post processing is possible by the user.
- Users submit jobs to the batch system under their username.
- If a hacker compromises an application, they would be running under a username with no extra permissions.
- PUDL supports in-place sharing of common resources such as shared libraries and themes, allowing us to avoid duplicate installations of these elements.

3.2.3 Template customization and integration with framework

The final step in our SPLE process is to develop new portals under the OnDemand store. This is accomplished by customization of the PUDL template, building new pluggable components around it to cater to specific requirements of new portals and ultimately integrating it with the OnDemand infrastructure. This well-documented, step-by-step
process enables agile development of the new portal by bringing up a very basic version first and then further customizing it for specific needs. Ultimately the newly created portal is integrated with OnDemand’s central framework and made available through the OnDemand dashboard. The high level steps that enumerate the process of setting up a new portal are given below [14]:

1. Setup the Drupal databases for the portal and the user.
2. Install and enable the required modules.
3. Enable the pudl theme and set it as the default theme.
4. Modify the webforms to include portal specific parameters.
5. Modify the scripts for job creation, submission, parameter substitution, and status inquiry so that they reflect the correct portal and directory names.
6. Put the simulation scripts provided by the users in the appropriate directory.
7. Update the visualization handling code to invoke the proper 3rd party visualization packages/tools.
8. Add/modify/delete code to address portal specific features. For example, the Container Fill Portal required image visualization of results apart from default functionalities.
9. Complete OpenID authentication, External Programmable Interface (EPI) and MySQL database configuration processes that are required to add the new portal in the OnDemand’s catalog.
10. Commit the portal’s code to OSC’s Subversion (SVN) repository in order for it to be available through OnDemand catalog.

3.3 Generic SPLE process to build M&S application store

In this section we present a generic process to build application stores similar to OnDemand that offer HPC based M&S applications. This approach is extracted from our experience of developing the OnDemand store and its underlying applications. The process is illustrated in Figure 7 and explained subsequently.

Figure 7: Generic SPLE process for M&S Application store development
As Figure 7 shows, the process begins with feature modeling of all the target applications. Feature modeling provides hierarchical representations of feature requirements of individual applications. Once feature models for all applications are derived, every feature is mapped to solution domain modules that implement the corresponding feature. These features and their solution domain counterparts should also be categorized into common and application specific. Common requirements in HPC context could be interface themes, job/task management scripts, database management modules and system monitoring interfaces. Application specific requirements could be visualization tools to be used, web-interfaces, user parameters, run scripts, software and hardware dependencies. The common components in the solution domain must be designed and implemented to work cohesively and construct a framework that could be shared by all applications. We divide this entire infrastructure into two categories: App Kit and App Runtime, as explained below.

1. **App Kit**: The purpose of the App Kit is to enable developers to create applications that use HPC resources. It should primarily consist of portable software modules that facilitate application development by third parties. We propose a multilayer architecture for App Kit consisting of a web access layer, a web application layer and a system interface layer. The web access layer should ensure authentication (preferably single sign-on) and isolated workspaces to HPC users. The web application layer should provide reusable APIs and application templates written with
industry-standard technologies such as OnDemand’s PUDL template for new application creation. Finally, the system interface layer should provide direct access to local resources such as databases, file systems and HPC job queues. Having separate layers for specific goals would greatly reduce the application development time and effort. Our proposed App Kit is illustrated in Figure 8 and should be comprised of the following components:
Figure 8: App Kit components
a. **Web Interface components**

App Kit should contain a set of components that can be customized by application developers to create user-friendly web interfaces. This includes web themes, web-forms layouts, branding mechanisms, interfaces for common job operations such as view, edit, delete, duplicate, visualize etc. These components are typically written in HTML, css, JavaScript, JQuery, php and other web programming languages. Appendix A shows some interfaces of OnDemand’s applications that are built on such common web modules integrated within the PUDL template.

b. **PBS scripting components**

PBS scripts are written in a specialized language and contain the directives that launch the HPC job(s). Their usage has a great impact on job efficiency. App Kit should contain PBS script templates and guidelines for writing advanced scripts. A PBS-specific DSL would make the most-common PBS directives easier to specify safely. A sample PBS script snippet is given below:

```bash
#PBS -N ContainerFill
#PBS -l nodes=8:ppn=12
#PBS -l walltime=65:00:00
#PBS -q @oak-batch.osc.edu
#PBS -j oe
#PBS -S /bin/bash
cd $PBS_O_WORKDIR
```
c. **Staging process components**

Most simulations operate on data files - a set of files, directories and possibly databases that contain run parameters. A staging process is needed to set up these data files correctly and insert the input values into them. App Kit should provide basic templates for **copying files and directories** and **parameter substitution**, and enable developers to write their own staging scripts in their preferred scripting language. A sample code snippet written in perl that handles job submission is given below:

```
$qsub_command='PBS_DEFAULT=oak-batch /usr/local/torque-2.4.10/bin/qsub'
if($? != 0);
my $subed = `$qsub_command runScript.txt 2>&1`;
exit 126 if($? != 0);
...
```

d. **Application environment configuration components**

Web applications may intend to set up or customize their environments at runtime. Our target web applications work on Apache web server’s environment. Apache configurations are complex and sensitive to incorrect settings. Therefore limited configurability should be allowed in the Apache’s
runtime settings. App Kit should include a restricted set of components to minimally configure runtime environment through “.conf” files and Linux modules, based on application specific requirements.

e. Simulation lifecycle management components

Any typical simulation goes through the three phases of creation, execution and post-processing. App Kit should include modules to deal with each phase distinctly. OnDemand’s PUDL template includes several distinct modules to handle these phases.

- The job creation phase should be managed by web-forms to take user inputs for simulation parameters and components that store and retrieve those parameters into databases or other persistent medium. The same components should also handle view, edit, delete, and duplicate operations.

- The job submission phase should be managed by data file allocations and parameter substitution components followed by PBS scripting and status reporting modules.

- The post-processing phase should be managed by post-processing scripts followed by results verification and ultimately invoking visualization tools or web-interfaces.
f. **Real-time feedback components**

Web applications are expected to be almost instantaneously responsive while HPC applications may take a prolonged time to complete a task and produce output. In the course of task execution, real-time consumption of system resources (per node memory, CPU, IO, network etc.) is subject to be monitored and their status should be reflected back to web interfaces. App Kit should have system monitoring modules that monitor and notify users so that undesirable situations such as deadlock, starvation, node failure can be addressed by the appropriate entities. OnDemand’s **job monitoring** interfaces shown in Appendix A, exhibit this functionality.

g. **External programming interface components**

In order to facilitate third parties for web applications development, App Kit should provide an external interface other than a GUI. HPC applications are not often publically available; instead they are often accessible to only certain user sections. So their accessibility must also be governed through providing back-end interfaces to third parties. For example, in the current OnDemand’s framework applications are cataloged by a certain vendor that enables access to OnDemand and its applications through OSC’s external interfacing mechanism described in section 3.1.
h. Accounting and billing

To establish a pay-as-you-go model, App Kit should provide different flexible accounting and billing schemes to users. One traditional approach is to bill users based on compute cycles usage for various HPC services; another could be based on maximum resource quota allocation for each user per application. Some of OnDemand’s applications are distributed per user license and cost estimation is done based on overall system resources utilized. There are many other accounting and billing models available for similar services that can be adopted and customized through App Kit.

2. App Runtime: App Runtime should include the software and hardware configurations that are not application specific but should provide the necessary environment for all applications to run. It should be responsible for launching web applications on the cloud upon request through application store. OnDemand Runtime (ODR) provides similar functionalities to support provisioning of its applications. App Runtime should also feature a multilayer architecture. The top layer should manage configuration for authentication. The middle layer should implement web protocols for the web application server. The bottom layer should provide a system-level interface for creation of dynamic services and access to system resources. App
Runtime should create a “system abstraction layer” allowing smooth integration with new systems such as Amazon EC2 [2].

One important aspect for Runtime to control is **load balancing and system recovery**. HPC users may not always be knowledgeable about the patterns of system resource consumption by their jobs. There may be situations where due to erroneous implementation of programs, the system experiences imbalanced resource utilization or even worse deadlock. Rather than manual intervention, Runtime should be able to resolve such situations, identify responsible jobs and apply resource throttling to recover the system to normal state. For example, having a default upfront quota on various resources based on job type might help users better program their jobs. Running low priority daemons to periodically check for prolonged hung or queued jobs and ensuring they reach to appropriate state would help the system recover faster.

In conclusion, we can infer that each application during their own development cycle must implement the application specific features in segregation by utilizing the common infrastructure facilities i.e. App Kit and App Runtime. Note that App Kit and App Runtime should not only facilitate in the application development process, but also
ensure non-functional requirements such as the integrity, isolation and security of web-applications and user data.

In this chapter we presented an overview of the OnDemand application store, its architecture and the SPLE approach followed to build it. We also extracted from our experience a generic process to build similar application stores. In the next chapter we focus on the validation of our approach by demonstrating the developmental outcomes and benefits of specific portals. We also present quantitative and qualitative analysis of our approach through measurable and non-measurable (intuitive) benefits and the challenges faced.
CHAPTER 4: RESULTS AND ANALYSIS

OnDemand and the encompassed applications (M&S and system utilities), including under development ones, have resulted from practicing systematic approach of software product line engineering over a limited period of time. SPLE approaches have largely improved the productiveness and in turn the methods of service provisioning at OSC. We now focus on validation of our approach through demonstrating quantifiable and non quantifiable benefits. We first discuss front-end level benefits through comparing user interfaces of Ondemand and its applications, particularly Container Fill Portal (CFP) and SA Weld Predictor (SAWELD). We then establish quantitative and qualitative analysis of our approach. We finally discuss problems and challenges encountered in following the SPLE approach.
4.1 User interfaces of OnDemand and its portals

Figure 9 shows the primary OnDemand Dashboard interface. The user is availed various options here such as access to HPC file system through file browser (via open source AjaXplorer project http://ajaxplorer.info), job creation and system monitoring interfaces where users can create, submit and monitor jobs via web, terminal access to OSC’s clusters, and a variety of M&S, utility, and visualization applications. Note that, due to the unique dual authentication levels (web and UNIX) in OnDemand’s architecture, users are permitted to access and manipulate their file system via web. More screenshots of OnDemand interfaces are provided in Appendix A.

![OnDemand Dashboard](image)

**Figure 9: The OnDemand Dashboard**
As mentioned before OnDemand’s Dashboard is a unified entry point for users to OSC’s HPC services. Following SPLE helped in integrating various utility and M&S applications under one interface and over a common framework. This largely improved user experience by eliminating the need to work around segregated means to access and use HPC resources. A distribution of applications for January and February 2013 is presented in Figure 10 [13].

![Figure 10: Application Utilization by Category](image)

Figure 11 shows the primary interface of CFP portal. This portal is used to simulate variety of fluid’s filling operation in massive scale containers. CFP is one of the National Digital Engineering and Manufacturing Consortium (NDEMC) [32] projects. NDEMC is one of OSC’s major collaborators and user of HPC services through portals.
for many of its projects in different domains. The interface shows variety of operations such as new simulation creation, view, edit or delete previous simulations, submit and visualize. These are standard set of operations that most of the M&S portals provide to users. While features such as view STL and view results are portal specific requirements and implemented through customizations and new modules. Figure 12 shows the new simulation constructor page, which also requires customizations in web form component of PUDL template to include simulation specific parameters.

![Figure 11: Primary interface of CFP](image-url)
Figure 13 shows the primary interface of SA Weld Predictor (SAWELD) portal. This portal is used for welding process simulations. It is noticeable that the user interface themes and standard set of operations are very much similar to that of CFP. This indicates uniformity in features implementation (achieved through software reusability of PUDL template) and benefit of avoiding redundant implementations. However portal specific features, web-forms, simulation scripts, and result visualization mechanisms are ought to differ among portals as they are the specific requirements of the corresponding portals.

Figure 14 shows the new simulation constructor of SAWELD portal which significantly differs to that of CFP due to difference in simulation specific parameters. However such differences are managed through simple customizations in the PUDL template.

Figure 12: New job constructor interface of CFP
Figure 13: Primary interface of SAWELD portal

Figure 14: New job constructor interface of SAWELD portal
Screenshots of some more portals are provided in Appendix A. Other M&S portals also share similar architecture, user interfaces and implementations of common features through reuse of PUDL template and OnDemand’s foundational framework and avoiding heterogeneity in feature implementation. This consistency in web lay-outs results in better user experience while software reuse and homogeneous feature implementations result in better developer’s experience.

4.2 Validation of SPLE approach in building OnDemand

The essence of SPLE methodologies was primarily maximizing software reuse through common framework and PUDL template. It had significantly optimized development time and effort while building OnDemand and its portals. We now validate our methods through first quantitatively comparing development time and effort for various portals, and then presenting the qualitative benefits of our approach.

4.2.1 Quantitative analysis

The basis of organizing framework components and building PUDL was formed after developing the primary portal Manifold Flow Predictor (MFP). It took a significant amount of time to write, primarily because it was the first portal developed under Per user Web Server architecture. Subsequently Heat Sink Portal (HSP) was derived from MFP portal’s code base. Since the backend requirements of all portals were similar, it was
sensible to first extract the infrastructure functionalities (PUWS, OnDemand Web Service, OnDemand Proxy, and SSO) and organize them as framework components, and then extract the essential and reusable modules from initial portals and construct the PUDL template with them. Future portals have been using this entire infrastructure and largely benefitted. Table 3 summarizes the chronological order of development of portals. Note that after developing first portal, the same developer saves the time spent in learning the new framework and template, while developing subsequent portals, thereby improving the overall turnaround time to develop multiple portals. The reduction in development time and new code establishes the benefits of SPLE approach.
<table>
<thead>
<tr>
<th>Portals</th>
<th>New Lines of Code excluding the template code (Approx.)</th>
<th>Time taken and % reduction with respect to MFP (Approx.)</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFP</td>
<td>3000</td>
<td>10 weeks (0% reduction)</td>
<td>Completed prior to the PUDL template.</td>
</tr>
<tr>
<td>HSP</td>
<td>2700</td>
<td>3-4 weeks (60-70% reduction)</td>
<td>Derived from the MFP code base.</td>
</tr>
<tr>
<td>PUDL based template and documentation</td>
<td>2000</td>
<td>2 weeks (Not applicable)</td>
<td>Integrated the common elements of MFP and HSP.</td>
</tr>
<tr>
<td>OnDemand Framework and Interface</td>
<td>3000</td>
<td>4 weeks (Not applicable)</td>
<td>Partially derived from MFP and HSP. Development by group of 3-4 developers.</td>
</tr>
<tr>
<td>CFP</td>
<td>400</td>
<td>3 weeks (70% reduction)</td>
<td>Includes time to understand the template and modifications.</td>
</tr>
<tr>
<td>SAWELD</td>
<td>600</td>
<td>4 weeks (60% reduction)</td>
<td>Created by the CFP developer.</td>
</tr>
<tr>
<td>Job Constructor</td>
<td>300</td>
<td>2 weeks (80% reduction)</td>
<td>Created partially by the CFP developer.</td>
</tr>
<tr>
<td>ACP</td>
<td>1000</td>
<td>5-6 weeks (40% reduction)</td>
<td>Derived from MFP with feature additions.</td>
</tr>
<tr>
<td>FAN</td>
<td>2500</td>
<td>8-10 weeks (5% reduction)</td>
<td>Low number of common features; greatly varies from other applications models.</td>
</tr>
</tbody>
</table>
4.2.2 Qualitative analysis

From users’ and developers’ experiences, we identified the following non-quantifiable benefits that support effectiveness of our approach.

i. OnDemand was aimed to provide unified interface to access the HPC resources and utilities. This eliminated the barrier for new users in exploring M&S through HPC techniques, while avoiding the need to learn low level system usage techniques and helped them focus on execution, monitoring and visualization of simulations through better interfaces.

ii. The dual authentication of ODR-user (web and OS) provides higher flexibility and security to users. Firstly, it allows users to access, upload, download and manipulate their data at OS level (as OS-user) even after logging in through web (as web user). Secondly, it allows system-level security mechanisms to operate on system processes launched from the gateway [13]. Finally, it prevents user data from intrusions because compromising the web server/application only enables access to file system of same web user, without privileged access to the system, neither to other users data.

iii. PUDL framework was built by first following an extractive approach [42] i.e. extracting common features from initial portals, then following a reactive approach i.e. customizing the template for specific features of new portals and progressively refining the template.
iv. The PUDL template was reusable and easy to understand. Specially, after understanding the template and building one portal using it, the learning time overhead decreased and development turnaround time was also improved.

v. Use of the template support concurrent development resulting in higher throughput and productivity.

vi. A common code base facilitates collaborations between developers, whether simultaneously or over time. These collaborations aid in managing developer transitions and improving code quality and maintainability.

vii. The template provides consistent layouts and features across portals. This consistency allows users to understand navigation and layout of new portals after becoming familiar with first portal.

viii. The OnDemand framework and PUDL template decreased the time required to test and debug portals, since the template was well tested.

ix. The OnDemand framework and PUDL template simplified branding and decreased time to market.

x. The OnDemand framework and PUDL template improved the code maintainability to a great extent. Due to common implementation of similar features, it became very convenient for developers to add common features or make corrections to multiple portals simultaneously.
4.3 Problems/challenges faced in our approach

Despite the aforementioned benefits, a few potential limitations to our approach were also identified in the course of developing OnDemand and underlying portals:

1. Portal specific customizations must be addressed individually, which may significantly influence portal development time. For example, development of Horton Fan Portal (FAN) (a comprehensive portal to simulate fan models; screenshots provided in Appendix A) was prolonged due to specific feature requirements particularly in front-end, despite being benefitted from OnDemand framework.

2. Templates are not applicable to all portals. Developers must evaluate whether a template is relevant to their case, or their portal may not realize the benefits of using a template e.g. FAN portal rarely used the PUDL template for its feature implementation.

3. A common problem with the scalability of the product line approach is that, an increase in the number of variant features and feature dependencies often leads to an increase in similar component versions. These sow confusion and hinder reuse [14, 42]. We restricted module versions by identifying similarity patterns in those modules and refactoring them into generic modules. For example, in each portal, rather than duplicating similar code for invoking 3rd party visualization packages (e.g. ParaView, Abaqus), we created a separate library which served each portal’s requirements.

4. A system level limitation was that the unconventional practice of running one web server per user consumes more RAM than a single web server [13]. In OnDemand’s
architectural design, each http request to a PUWS goes through the OnDemand Proxy. However, initial studies of the RAM utilization demonstrate the overhead of additional httpd instances is within a factor of five to ten compared to sshd. Stress tests have been conducted with as many as fifty concurrent PUWS on a single web service node. If a user neglects to log out an external process will periodically monitor for inactivity to shut down inactive PUWS.
CHAPTER 5: FUTURE WORK AND CONCLUSION

5.1 Future work

There is a great scope of enhancements in multiple areas of OnDemand as future work. We will now discuss two areas that would elevate the degree of SPL’s application and also greatly impact OnDemand’s reach to users and developers communities in HPC and M&S areas.

5.1.1 App Kit and App Runtime

We discussed about App Kit and App Runtime in section 3.3. We aim to provide these two components to engineering service providers, software companies and cloud providers to rapidly and efficiently develop applications for new markets. App Kit is aim is to facilitate application development by third parties targeted to OSC’s HPC environment. While App Runtime’s purpose is to abstract away the system level functionalities such as authentication, resource administration and so on that are common to most applications and facilitate the application deployment in OSC’s environment.
5.1.2 Automation framework for portal generation

While developing the portals using the template, we realized that although the template provided basic functionalities, portal-specific functionalities still remained to be implemented. In particular, each portal required simulation specific parameters and these parameters required special handling based on simulation scripts. This required code instrumentation at deterministic places in the template to generate the portal’s code. For example, all the portals need to create key-value pairs in database to store simulation parameters. This instrumentation is a tedious and error-prone task. Therefore it would be wise to automate this task. There are many other similar changes that can be automated through a code instrumentation suite.

We propose an automation suite that would take the portal’s requirements from users through an interface and generate the basic portal on the fly, using the current template, without a programmer’s intervention. Figure 15 represents the high level architecture of our proposed suite. More enhancements are possible to this suite that might further address custom features such as job duplication, server file retrieval etc. This suite would reduce the developer’s time and effort and bring end-users into the portal construction and customization process, thus enable End-User Computing in OSC’s environment and improve the overall user experience. We intend to work on interface design and implementation of this automation suite in near future.
Figure 15: Automation suite for building web portals
5.2 Conclusion

OSC OnDemand is a unified web platform where users can perform system level HPC operations through web based applications. Most of these applications are particularly used by researchers, scientists, and programmers for HPC oriented M&S. The idea behind building OnDemand was to align the segregated mechanisms of accessing HPC resources and provide users with a comprehensive product for their HPC requirements. We tried to facilitate agile development of OnDemand and its applications through a cycle of feature modeling and software product line engineering.

In this thesis, we first explained the importance of M&S based design and manufacturing and HPC’s role in accomplishing it. Then we did the case study of HPC based M&S applications followed by Software Product Line Engineering and Feature Modeling; and showed how they pertained to our context. We began the solution description by describing the OnDemand and its architecture at a high level. We then explained the Feature Modeling and SPLE processes followed in developing the common framework, the PUDL template and finally the portals under OnDemand. We concluded with extracting a generic SPLE process from our experience to build similar application stores as OnDemand.

59
We began demonstrating the outcomes of our approach by showing the interfaces on OnDemand and various portals and indicating the commonalities and variations among them. We discussed how effective the approach was in terms of time and effort and other qualitative benefits. We also mentioned some of the problems and challenges we encountered during the development of OnDemand and the portals. We finally proposed an automation suite to build portals more efficiently. This automation is intended to involve users in the portal construction and customization process and to generate portals on the fly with user input. This would introduce end-user computing into our environment and reduce the efforts of programmers. We intend to build this suite in the near future to establish a better process for portal development.
REFERENCES


[16] Drupal website: http://drupal.org/home

[17] Federated Login for Google Account Users:

https://developers.google.com/accounts/docs/OpenID


http://www.cs.colorado.edu/~kena/classes/5828/s12/presentation-
materia
dibieogheneovohanghaojie.pdf

[22] Hudak, D., Stredney, D., Calyam, P., Wohlever, K., Bitterman, T., Hittle, B.,
Kerwin, K., and Krishnamurthy, A., *Enabling Data-Intensive Biomedical Science:
Gaps, Opportunities, and Challenges*. OMICS A Journal of Integrative Biology,
15(4) 2011.

[23] Hudak, D., Bitterman, T., Ramnath, R., Calyam, P., Ramanathan J. and Zhang, D.,
Simulation as a Service: *A Cloud-Based Framework to Support the Educational
Use of Scientific Software*. The 1st International IBM Cloud Academy Conference


Technical report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie
Mellon University, 1990.

nanoHUB.org: *Advancing Education and Research in Nanotechnology*. Computing


[34] Ohio Supercomputer Center (OSC) website: www.osc.edu

[36] Overview for OpenID Planning:


[38] Rappture Toolkit: https://nanohub.org/infrastructure/rappture/


[40] San Diego Supercomputer Center (SDSC) website: http://www.sdsc.edu/

[41] Study of U.S. Industrial HPC Users,

European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering, September 05-09, 2005, Lisbon, Portugal.


[44] UNICORE Portal Task Force -

[45] VMWare’s End-user Computing Management:

[46] VMWare Horizon website:


APPENDIX A: SCREENSHOTS

A.1. OnDemand Interfaces

A.1.1 Login Page

URL: https://ondemand.osc.edu/catalog/

Credentials: OSC’s HPC credentials are required.

Figure 16: OnDemand’s Login Page
A.1.2 The DashBoard

Figure 17: OnDemand’s DashBoard
A.1.3 The AjaXplorer File Browser

Figure 18: The AjaXplorer file browser
A.1.4 The Cluster Terminal

Figure 19: Cluster terminal
A.1.5 The Job Constructor

Figure 20: The Job constructor
A.1.6 The Job Monitor

Figure 21: The Job monitor
A.1.7 Apps Menu

Figure 22: Apps Menu
A.2 FAN Portal Interfaces

A.2.1 Project Viewer

![FAN Portal’s Project Viewer](image_url)

**Figure 23:** FAN Portal’s Project Viewer
A.2.2 New FAN Simulation constructor

Figure 24: FAN Portal’s New simulation constructor
A.3 Weld Predictor Portal interfaces

A.3.1 Primary interface

Figure 25: Weld Predictor Portal’s primary interface
A.3.2 New Weld Simulation constructor

![Figure 26: Weld Predictor Portal’s new simulation constructor interface](image)

Figure 26: Weld Predictor Portal’s new simulation constructor interface
A.4 Truck Add-on Predictor Portal interface

A.4.1 Primary interface

Figure 27: Truck Add-on Predictor Portal’s primary interface