Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2007 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2007

Achieving Flexibility via Service-Centric Community Source: The Case of Kuali

Manlu Liu

Harry Wang
University of Delaware

Leon Zhao
University of Arizona

Leon Zhao
University of Manitoba

Leon Zhao
University of Arizona

See next page for additional authors

Authors
Manlu Liu, Harry Wang, Leon Zhao, Leon Zhao, Leon Zhao, and Leon Zhao

Achieving Flexibility via Service-Centric Community Source: The Case of Kuali

Manlu Liu[1]
Department of MIS
University of Arizona
manluliu@arizona.edu

Harry J. Wang
Dept. of Accounting & MIS
University of Delaware
hjwang@lerner.udel.edu

J. Leon Zhao
Department of MIS
University of Arizona
jlzhao@u.arizona.edu

Abstract

Service-centric application development has become one of the most important paradigms in software engineering, and collaborative open source development has emerged as a new way for organizations to develop new applications that are either not available or too expensive to acquire. A most recent trend is to combine service centricity with open source development collaborated by multiple organizations to develop scalable and flexible information systems. However, service centricity and open source collaboration involve innovative concepts and practices that are very new to many. As such, there are a lot of confusion among both researchers and practitioners in this area. In this paper, we investigate several important concepts surrounding service-centric open source development via the case of Kuali including service-oriented architecture, workflow technology, and technology flexibility as well as the relationships among those concepts. In particular, we concentrate on understanding the relationships between technology flexibility and emerging technologies. One important contribution of this paper is the introduction of community source as a unique open source concept to the MIS community. Finally, we report several research findings and outline a number of research directions in the related areas.

Keywords: *technology flexibility, open source, community source, business process management, workflow technology, service-oriented architecture, service centricity, service-centric development*

Introduction

Organizations are facing an environment of ever increasing turbulence and change on a global scale. To survive under global competition, they need to meet more and more challenging requirements, such as shorter product cycle times and continuous costs reduction. One way to meet the new challenges is to improve the flexibility of enterprise systems, thus making the organization more adaptive and agile to unexpected changes. In particular, increased flexibility can give an organization more competitive advantage through faster response to varying customer needs and environmental conditions. As companies automate their business processes through workflow technology and electronic business, the flexibility of business process implementation can also greatly influence the organization's capacity for change. As has been long recognized by researchers (Nelson et al. 1997), companies have been using emerging technologies to enable flexibility in the enterprise computing environment. Most recently, a new set of information technologies collectively referred to as service-oriented computing have been adopted to support more enterprise agility (Zhao, Tanniru, and Zhang, 2007).

In this paper, we examine the importance of service-centricity in open source development, in the context of Kuali, a collaborative project among nine major universities in the US (www.kuali.org). Kuali is an ideal case for studying service-centric open source development. It started with a focus on developing a university financial information system and has now extended the system scope by including research administration and student systems. The goal of Kuali is to combine resources from multiple universities to minimize development risk and financial burdens. Kuali is developed as an alternative to expensive and rigid systems such as Financial ERP modules offered by commercial software vendors. Kuali has been referred to as community source because it is an open source project developed by a community of functional managers and systems developers from nine major universities collaboratively. The community source project is being developed through several phases. All adopting organizations can customize the Kuali system, but the source code must be properly documented and shared by the whole community.

[1] Ms. Manlu Liu is also a faculty member at Zhejiang University and is currently on leave while pursuing her doctoral study at the University of Arizona.

As a community source project, Kuali is a unique form of open source in which the community must balance the various, sometimes even conflicting, requirements from all development partners. As such, community source is more challenging than a general open source project in which variations in requirements are not a major issue. In our view, community source has more stringent requirement in flexibility in order to deal with complex requirement analysis and change management. Recently emerged technologies such as web services, service-oriented architectures, and workflow automation can help make a system more customizable. As we illustrate in the paper, Kuali takes advantage remarkably of these emerging technologies to enable the strong system flexibility. In fact, we believe that Kuali is a good representative of a recent trend in combining service centricity with open source development to build scalable and flexible information systems.

As we demonstrate via the Kuali case in a later section, the community source approach is a unique form of open source approach. There are several important distinctions between the two. First, community source requires a formal collaboration among multiple institutions via a virtual organization whereas the latter does not require so. Second, the application developers in a community source are employees designated to the project by the partner institutions. Third, the community source project requires significant initial investments by the partner institutions, but an ordinary open source project might not. Some of these three features might appear in other types of open source projects, but a community source project requires all of them. In summary, we define community source as a unique type of open source that depends on significant financial and other support from a community of institutions in the development and deployment of the enterprise application.

All in all, service centricity and open source involve innovative concepts and practices that are very new to many. As such, there are a lot of confusion among both researchers and practitioners. In this paper, we investigate the concepts and issues surrounding these two important topics based on the recent literature. Further, we examine the case of Kuali and its technological components, including service-oriented architecture, document workflow, and customization techniques.

The structure of the paper is as follows. First, we review the recent research in technology flexibility, open source development, workflow technology, and service-oriented architecture. Second, we introduce the Kuali project by discussing its project goal, organizational structure, and its system architecture. Third, we describe the five levels of customization in Kuali and explain how Kuali achieves flexibility via various software techniques. Finally, we discuss our research findings by means of several propositions and outline a number of research directions stemming from the Kuali case.

Literature Review

As explained above, our focus in this paper is to illustrate how to achieve flexibility under service-centric community source. Next, we summarize the recent thinking in several related research areas, including technology flexibility, open source development, business process management, and service-oriented architectures.

Technology Flexibility

Flexibility has emerged as a key competitive factor in many organizational endeavors. Nelson et al. (1997) defines technology flexibility as "the ability to adapt to both incremental and revolutionary change in the business or business process with minimal penalty to current time, effort, cost, or performance." A measurement framework for technology flexibility was proposed including such factors as modularity, change acceptance, and consistency in the structural flexibility dimension, and rate of response, expertise, and coordination of actions in the process flexibility dimension. Zhao (1998) differentiated between two software flexibility concepts: system adaptability and system versatility. The former is defined as the capability to modify the system to cope with major changes in business processes with little or no interruption to business operations, and the latter is the capability of the system to allow flexible procedures to deal with exceptions in processes and procedures. Deiters, Goesmann, and Löffeler (2000) dealt with the issue of flexibility in technical, organizational, and human perspectives. They classify flexibility into four dimensions: process flexibility, inter-organizational flexibility, flexible management and knowledge, and flexible task allocation. Byrd, et al. (2000) focused on the IT infrastructure flexibility and attempted to develop a valid and reliable instrument for IT infrastructure flexibility.

Our work in this paper extends the literature by examining the relationship among three related characteristics, namely technology flexibility, system extensibility, and software customizability. Essentially, we find that technology flexibility is a continuum of two stages of implementation. At the lower level, software can be customized without making major changes to the system code, leading to software customizability, and at the higher level, the system can be extended by either modifying the code or by adding new components. The key issue is about how emerging technologies such as service centric computing and business process automation can facilitate those capabilities.

Open Source

Open source has become an important software engineering methodology in which the source code of a system is made available to its adopters (Ducheneanut 2005). However, open source is a very complex concept because of variations on how it is developed, marketed, and adopted. The most well-known example is the Linux system that is developed and maintained by a global network of volunteers and distributed by some commercial companies such as Red Hat (www.redhat.com). Many versions of Linux are distributed currently by these commercial companies who generate revenue by adding new features and providing consulting services. Many other open source systems are developed with government funding as output of a research project and given online for free adoption with or without technical support (www.sf.net).

As a research domain, open source has received a lot of attention from various researchers, particularly the communication issue. Curtis et al. (1988) found that large projects have extensive communication and coordination needs that are not mitigated properly by documentation. These collaborative problems are exacerbated when the development team is geographically or organizationally distributed, as is becoming increasingly common. More effort is required for more systematic interactions when participants are distant and unfamiliar with each others' work (Ocker and Fjermestad 2000). Kraut and Streeter (1995) found that the formal and informal communication mechanisms for collaborative work on large-scale, complex software projects are important for sharing information and achieving collaboration in software development. Stewart et al. (2006) developed a framework of the main tenets of the open source software (OSS) ideology, which impacts the team effectiveness in various ways. Recently, Fitzgerald (2006) found the field of open source has undergone a major transformation from its free software origins to a new form of community-based development and adoption.

In this paper, our study about Kuali calls for new concepts and ideas on collaborative open source development among multiple institutions. Many research issues need to be explored. For instance, the community source in Kuali is based on a consortium of nine major universities. It is not clear if this paradigm would work well in a consortium of commercial companies. If so, what would be the difference in both development and adoption?

Workflow Technology

Workflow technology has been widely adopted by organizations to automate business processes, reduce cycle time, and support e-business. Workflow Management Systems (WFMSs) remove the process dependency by separating the process logic from the application logic, which makes information systems more flexible and adaptive to process changes (Stohr and Zhao 2001). Many approaches have been proposed to make WFMSs even more flexible. Casati et al. (2000) proposed a rule-based workflow modeling method to allow a high degree of flexibility during workflow design and great adaptability in terms of process exception handling. Document-driven workflow is another such effort, which is different from production workflow systems (Wang and Kumar 2005). The latter models the workflow as an execution sequence of tasks while the former designs a process by specifying the data dependences among the documents. As a result, document-driven workflow is very suitable for less-structured, cooperative, and ad hoc workflows.

Today, many companies are investing a great amount to expose their business operations as web services in order to offer easy integration with a wide range of third-party applications. Web services choreography and orchestration technologies have been developed to connect individual web services to form higher value-added processes (Peltz 2003). Workflow technologies based on the workflow standards such as Business Process Execution Language (BEPL) have helped organizations achieve better process flexibility (Weerawarana 2005). As we will illustrate later, the Kuali project leverages workflow technology to enhance the flexibility and extensibility of the system.

Service-Oriented Architecture

One of the emerging technologies used in Kuali is service-oriented architecture, which enables better system extensibility. This is a critical feature that makes community source a viable development methodology. Service Oriented Architecture (SOA) was first proposed by Schulte and Natis (1996). They specified SOA as "a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes." SOA is commonly defined as a software architecture that allows the use of loosely-coupled software services to meet the needs of business processes and users (en.wiki.org). A service-oriented architecture defined by IBM contains four core components including business services, integration services, enterprise service bus, and infrastructure services, which work together to provide the capability for on-demand business (Keen et al. 2004). Typically, SOA consists of composable services, which are used to support the operations of a specific enterprise (Cherbakov et al. 2005).

Service-oriented computing is a broader term referred to as the computing paradigm that utilizes services as fundamental elements for developing applications (Papazoglou and Georgakopoulos 2003, Huhns and Singh 2005). Service-oriented computing is described by some researchers as the use of service-oriented architecture for implementing web services.

As indicated in the literature, service-oriented computing is process-driven. This is because business processes are a flexible way to organize and integrate enterprise applications at the business level (Zhao et al. 2005). At the computational level,

service orchestration is the main function of the enterprise service bus, the coordinator of the functional modules under the service centric infrastructure.

Service-Centric Community Source: the Case of Kuali

In this section, we present the case of Kuali and show how Kuali achieves flexibility via service-centric community source. Kuali is a consortium of universities to develop an open source financial service system, starting with the conversion of the Indiana University's Financial Information System to the web in 2004 (www.kuali.org). The original motivation of the project is that existing financial systems used in the universities are outdated and too difficult to maintain. The commercial products are often too expensive and hard to customize; some institutions paid tens of millions to companies for software and installation, but still need to build 15% of all the features they need themselves to handle specific financial transaction needs. Some schools and colleges find they need to operate expensive "shadow systems" to provide needed features that are absent in currently available ERP packages. The alternatives to buying off-the-shelf packaged financial software are equally daunting and can only be considered by the largest institutions. The Kuali project provides an attractive alternative to the "buy or build" dilemma. It pools institutional resources to develop an open source financial system, thus dramatically reducing the cost of managing fiscal data and processes in higher education.

Launched in August, 2004, the Indiana University and the University of Hawaii led the effort to build Kuali. In March, 2005, Kuali project got an award of $2.5M from the Andrew W. Mellon Foundation. In the same year, four new partners, Cornell University, San Joaquin Delta College, Michigan State University, and the University of Arizona, joined as core investors. In April 2006, Kuali Project announced the availability of the Kuali Test Drive, which enables institutions to explore Kuali Financial Systems. In July 2006, the University of California joined Kuali. In October, 2006, Kuali foundation released phase I. Phase IIa and Phase IIb are planned to be released in July 2007 and in July 2008, respectively.

Figure 1. Kuali Organizational Chart

Kuali uses a community source approach to allow member universities to share the cost for and the benefit from system development. Figure 1 shows the complexity of the organization structure. The development partners work in a project organization. The Kuali Board is the final decision maker during the development of Kuali. The Kuali functional council and Kuali technical council take care of the functional issues and technical issues of Kuali, respectively. The Extended board, Kuali investors, Kuali functional council, Kuali technical council and the project manager report to the Kuali board. The project manager supervises the project staff and development staff.

The initial mission of the Kuali consortium was to develop a baseline system for financial services and has now expanded to other systems such as research administation and student management. In order to ensure flexibility and extensibility, Kuali uses the most up-to-date approaches and technologies such as open source, workflow, and service-oriented architecture. Figure 2 illustrates the Kuali system architecture that shows how the enterprise service bus is used to support service-oriented architecture. Currently, there are three main application modules, financial services, research administration, and student system.

At the center of the Kuali system is the Kuali enterprise service bus (KSB), the core concept of service-oriented architecture that coordinates the interactions between an application module and the various lower-level system services such as

workgroup service, user service, and legacy wrapper service. KSB includes a service registry, remote access protocols, and service invocation procedures. Each application module carries its own documents, business objects, business rules, data dictionaries, and other local services. This service-centric architecture makes it relatively easier to enhance the application modules and add new modules. As shown in Figure 2, Kuali also has a modularized middleware infrastructure called Rice, sitting between the server and applications. Rice consists of four client-side-components: Kuali Enterprise Service Bus, which includes a service registry, remote access protocols, and service invocation procedures; Kuali Nervous System, which provides reusable code and shared services to enforce system development standards; Kuali Enterprise Workflow, which facilitates the routing and approval of transactions throughout the university; and Kuali Enterprise Notification, which provides a single list for all university-related communications.

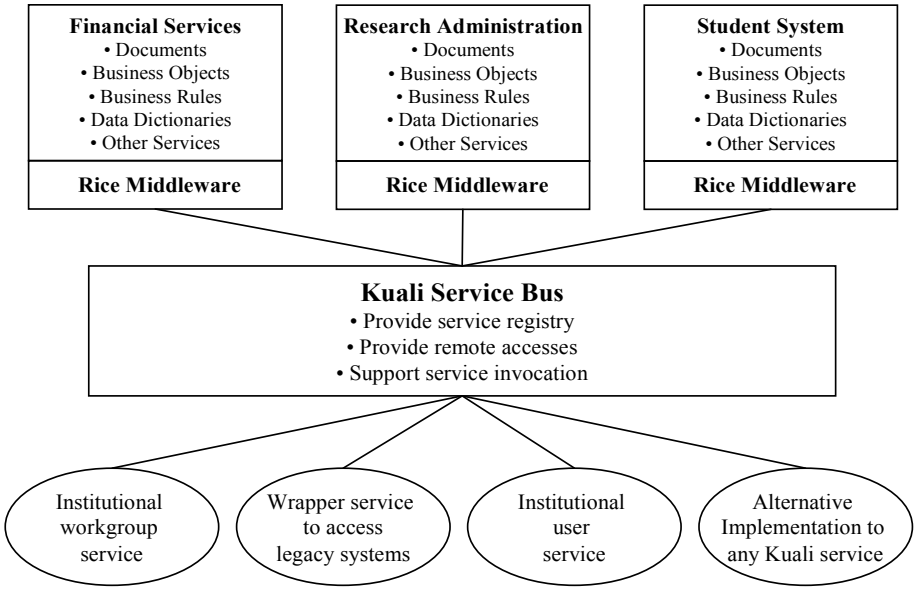

Figure 2. Service-Oriented Architecture of Kuali

Enabling Flexibility in Kuali

As a community source project, Kuali must balance among varying requirements from many universities. Thus, it is imperative to make the Kuali system flexibile in terms of both customizablility and extensibility. In this section, we take an in-depth look at how Kuali achieves flexibility in this unique environment.

Label Customization

An important aspect of flexibility is to enable the customization of the look and feel of its user interface. Specifically, Kuali allows its adoptors to rename any label on all forms (Figure 3) without the need to write code.

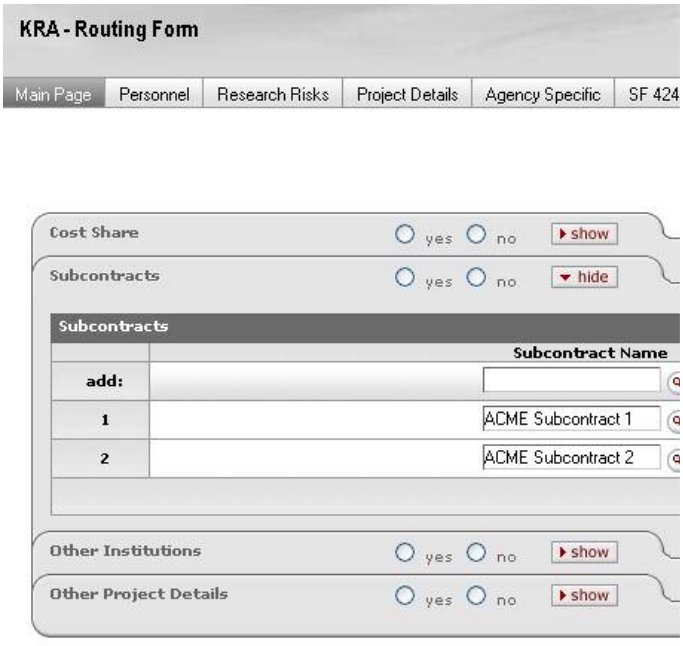

Figure 3. Kuali Forms and Labels

Kuali supports label customization by means of its Data Dictionary, which can be specified using XML files. For example, account.xml (Figure 4) holds various attributes about a Business Object "Account Number". This customizability allows an institution to change the user interface to fit its unique lingo without changing the lower-level programs.

```
<attribute name="accountNumber" forceUppercase="true">
```

```
<label>Account Number</label>  <shortLabel>Account</shortLabel> <maxLength>7</maxLength>
<validationPattern> <alphaNumeric exactLength="7" allowWhitespace="false" allowUnderscore="false" /> </validationPattern>
<required>true</required> <control>  <text size="10" /> </control>
<summary>Account Number</summary> <description>Identifier for a pool of funds.</description>
</attribute>
```

Figure 4. Data Dictionary Attribute Definition

Modification and Addition of Document Types

Kuali organizes its financial documents into a hierarchy of document types; as of today, it has 139 types on its website. Figure 5 illustrates a number of document types in the financial transactions category. Document types are a critical concept for Kuali since its business process evolves around the routing of documents.

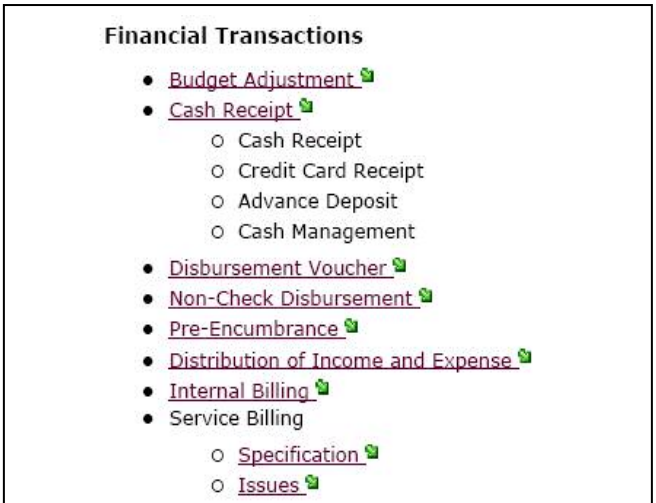

Figure 5. Kuali Document Type Examples

```
<dictionaryEntry>
    <maintenanceDocument>
        ...
        <businessRulesClass>org.kuali.module.chart.rules.AccountRule</businessRulesClass>
        <documentAuthorizerClass>org.kuali.core.document.MaintenanceDocumentAuthorizerBase</documentAuthorizerClass>
        <authorizations>
        <authorization action="initiate">
        <workgroups> <workgroup>kualiUniversalGroup</workgroup> </workgroups>
        </authorization>
        </authorizations>
        <documentTypeName>KualiAccountMaintenanceDocument</documentTypeName>
        <documentTypeCode>ACCT</documentTypeCode>
        <label>Account Maintenance Document</label>
        <shortLabel>AcctMaintDoc</shortLabel>
        <summary>Account maintenance document</summary>
        <description>Document used to create or update Account objects</description>
        <lockingKeys> <field attributeName="chartOfAccountsCode"/> <field attributeName="accountNumber" /> </lockingKeys>
        …
        </maintenanceDocument>
</dictionaryEntry>
```

Figure 6. AccountMaintenanceDocument.xml

Kuali allows its adopters to modify and add new document type via XML files. As shown in Figure 6, the Account Maintenance Document type is defined via a XML (only partially show for simplicity). The components of a document type include business rules, authorization, locking keys, and so on.

Workflow Customization

Workflows in Kuali can be customized to suit the needs of different universities. Essentially, Kuali workflow is document-based and allows the modification of document routing without having to write any code. As given in Figure 7, the Kuali workflow engine specifies the generic actions on financial documents including initial actions, approval actions, workflow processor, and post processor. Specific actions on each financial document are specified within the document type itself such as document initiator, organizational unit, and document approver. The Kuali workflow uses XML-driven routing capabilities, and routing rules can be defined or modified with respect to any document type without requiring a modification to the Kuali source code.

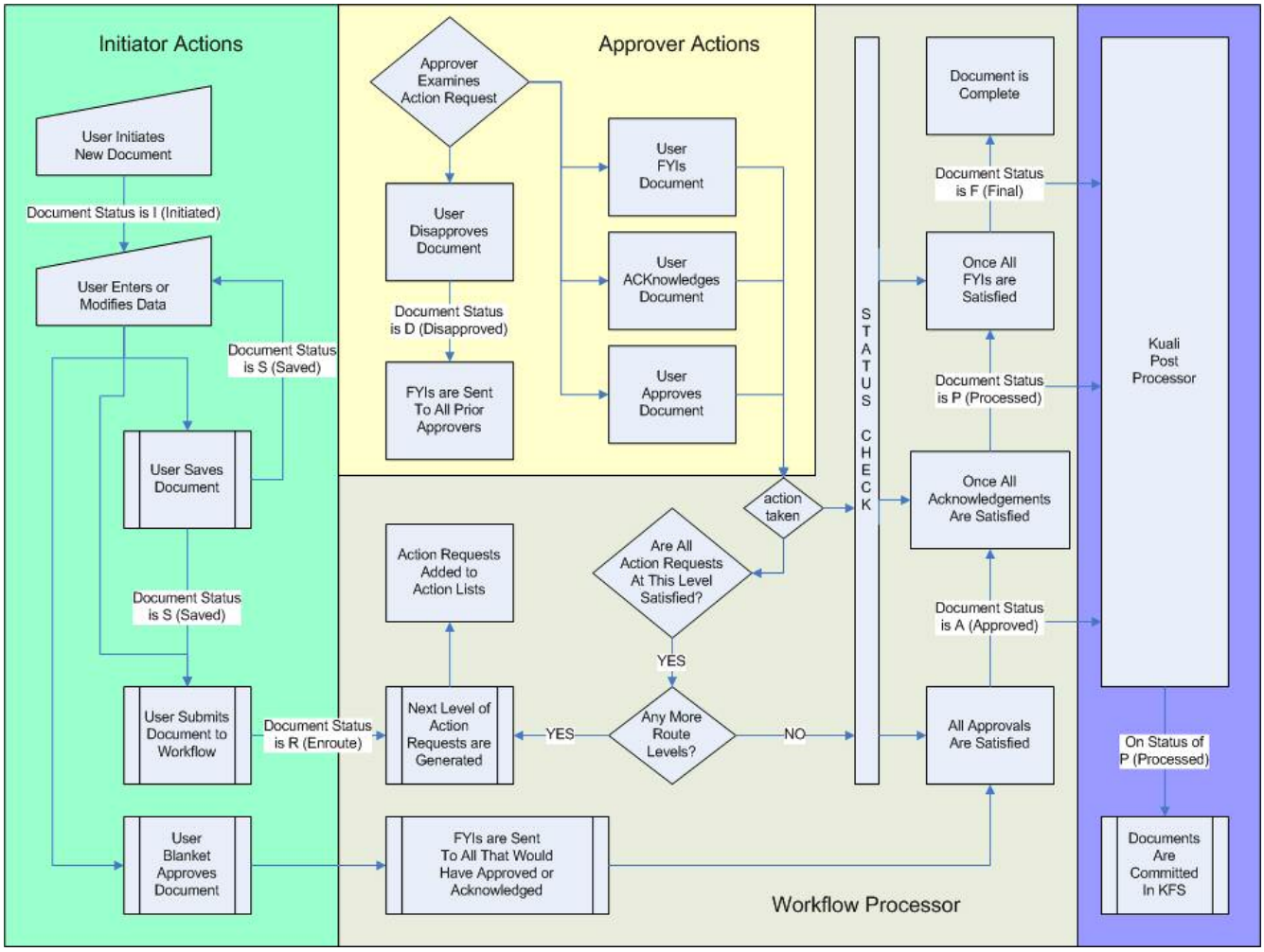

Figure 7. Kuali Workflow

Code Modification

This level of extensibility is inherent to the open source paradigm of Kuali. As an open source system, Kuali "allows its users to: 1) freely access, install, and run the software for any purpose; 2) modify the original software; 3) redistribute copies of the original or modified programs; and 4) share modifications with the community." Further, as a community source system, Kuali "is based on many of the principles of open source development efforts, but community source efforts rely more explicitly on defined roles, responsibilities, and funded commitments by community members than some open source development models."

Kuali is distributed according to the Educational Community License version 1.0, which grants the permission to use, copy, modify, merge, publish, distribute, and sublicense this original work and its documentation, with or without modification, for any purpose, and without fee or royalty to the copyright holder(s). The adopters are required to notify any changes or modifications to the original work.

Addition of New Modules

As discussed in an earlier section, Kuali started as a financial system and has evolved to an umbrella suite of administrative higher education systems. It does so by providing a common architecture that can be extended to incorporate new applications. The Kuali framework improves productivity and consistency across Kuali applications and creates a reusable development environment.

The service centric approach of Kuali makes it possible to "develop once and use anywhere and on any platform". This reduces the dependency of various development efforts and therefore simplifies the system architecture. Furthermore, workflow customization via business rules discussed previously makes it easier to create new application modules in a document-based routing. This further highlights the business value of service-oriented architecture, including the Kuali Rice Middleware approach.

Research Findings and Directions

The Relationship between Flexibility and the Community Source Approach

From the Kuali case, we can generalize several observations into a number of propositions, which will need to be validated in future research. Next, we state the propositions and explain them briefly without proofs.

Proposition 1: Service–oriented architectures enhance system extensibility.

As said previously, Kuali's system architecture is service-oriented. Its enterprise service bus provides the necessary facilities to enable various application modules to access enterprise services without being hardwired to these services. This helps the Kuali adopters extend the system by adding new modules while reusing much of these enterprise services such as user services such as user directories and institutional services such as data dictionary.

Proposition 2: Workflow automation enables model-based application development and system maintenance without having to revise lower-level code.

Model-based applications can be modified by revising the process models without having to change any program code. The Kuali case indicates that workflow can help model the routing and authorization tasks and make it easier for an organization to streamline its processes and tasks. This feature is more pronounced in a community base project like Kuali in which the system is designed with built-in flexibility.

Proposition 3: Community source requires more flexible software architectures in order to meet the needs of the community.

The Kuali case demonstrates the need for close and intensive interactions between the functional aspects and the technical aspects on the regular basis because varying needs of multiple institutions make it very difficult to get the specifications right. This is not as essential in other types of open source projects when the functional requirements are relatively stable.

To validate these propositions, we need to develop additional and more specific hypotheses that are based on measurable parameters. This will be done as part of our future research.

Research Directions

Next, we outline several research directions related to the Kuali case, including community source, workflow modeling paradigms, and software adoption methodology. We discuss unique research opportunities along with those research directions.

- **Community Source**

In a community source project similar to Kuali, software development teams acquire, maintain, analyze, and formalize software requirements from multiple stakeholders that can have varying and possibly conflicting demands. Different universities work together to develop the system, which may have conflicting demands. How to solve this issue and make the community based software development successful is an interesting research issue. In addition, effective collaboration is a major concern in large development teams. Designing complex software systems like Kuali requires effective communication among all shareholders. Although a few researchers studied the collaboration issue in software development, there are currently no formal guidelines and procedures that suggest to software and managers how to manage communications among all stakeholders of a complex software project. Successful community source projects must understand the political nature of complex software development and progressively enroll a network of collaborators to support their efforts (Ducheneaut 2005). Social network analysis can be an interesting research topic in the context of community source where the level of centralization may be correlated with project size and modularity (Crowston and Howison 2006).

- **Workflow Modeling Paradigms**

Currently, the Kuali platform specifies workflows by means of workflow rules in the context of financial documents. As such, these workflow rules are also called *routing rules*. These routing rules seem to work fine in the context of financial services since the workflows are relatively simple with typically three to five steps. However, it is not clear if routing rules would be appropriate in other Kuali modules. For instance, the Research Administration Module, currently under development, contains very complex workflows. The COEUS workflow (http://www.princeton.edu/~orpa1/coeus/) consists of over twenty processes, each of which could include many steps as illustrated in Figure 8. Specifying these workflows using routing rules could result in lengthy XML scripts that are very difficult to write and maintain. In addition, the control flow of those processes is not explicitly modeled when using rule-based and document-driven workflow modeling, which makes it difficult to understand the underlying process logic. This raises a research direction on when a particular workflow modeling paradigm should be used. Specific research issues include the development of criteria on how to choose between control-flow-driven workflow modeling and rule-based and document-driven workflow modeling (Wang and Kumar 2005), mechanisms and procedures to extract implicit control flow from workflow rules, and rule languages that are capable of modeling more complex workflows (van der Aalst et al. 2003).

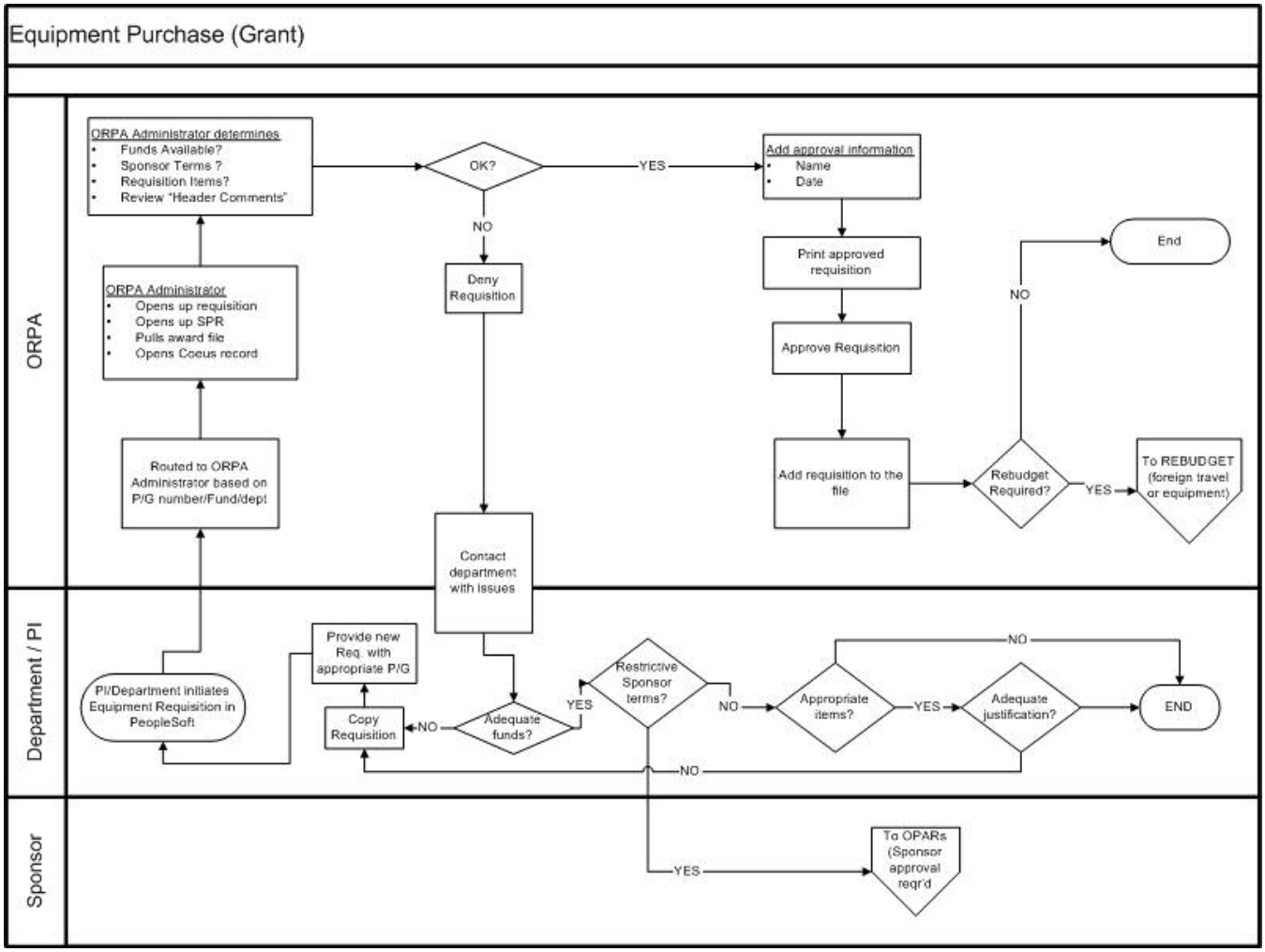

Figure 8. Equipment purchase process in research administration (http://www.princeton.edu/~orpa1/coeus/)

- **Adoption Methodology for SOA-based Software**

As demonstrated with the Kuali case, enterprises need to make software adoption decisions amidst recent technology advances such as service-oriented architectures and workflow automation. For example, how the universities other than Kuali development partners know Kuali is the right system or not for them is a non-trivial problem. As such, it is interesting to study how to describe the fitness of the system capabilities and enterprises needs. In order to study this issue, we need to understand how to formally specify the capabilities of the system, how to formally specify the needs of enterprises, and how to measure the fitness. Currently there is no formal methodology that allows enterprises to specify their system needs and

compare with software capabilities in a manner similar to conceptual database design via ER diagram. In other words, there should be a software adoption methodology to describe the fitness of the system capabilities and enterprises needs.

A related concept is business modeling (Cherbakov et al. 2005). Business modeling is an emerging area of research. Researchers have studied the difference between a business model and a business process model (Gordijn et al. 2000), how related project activities can be modeled consistently to support the reengineering effort of the same business entity (Gulla et al. 2000), and how to develop business models to support the development of complex systems (Petrovic et al. 2001). Our focus in this direction is to develop a methodology for adopting software systems for given business requirements.

Conclusions

In this paper, we studied the need for technology flexibility and how to achieve it in the context of the Kuali project, a multi-university and multi-year project by bringing an existing system to the web. We demonstrated how Kuali utilizes a service-oriented architecture and the community source approach to minimize project risk and maximize cost sharing in enterprise application development. At the core of the flexibility endeavor is the five types of customization in Kuali. It is clear that service centricity is the key ingredient in the Kuali recipe for achieving flexibility. In addition, rule-based workflow management, XML-based document specification, and the Rice middleware approach are necessary ingredients as well. Based on the Kuali case, we summarized our observations in the form of research propositions and directions. To the best of our knowledge, this paper is the first article examining the relationship between technology flexibility and service-centricity. Another important contribution of this paper is the study of community source in the context of a large-scale project.

We plan to tackle some of the research directions given in the last section. We will monitor the development and deployment of the Kuali system and continue with the research ideas we have presented in this paper. Validation of the propositions we outlined in this paper is another future research we plan to pursue.

Acknowledgements

We would like to thank Brad Wheeler, Elizabeth Taylor, and Kymber Horn for inviting us to attend the Kuali Days Conference held in November 14-15, 2006 and for providing valuable information about Kuali. We would also like to thank Brian McGough and Andrew Hollamon for providing us with in-depth technical knowledge about Kuali. The third author's research is supported in part by the 2005 IBM Faculty Award on Services Computing and Business Process Management.

References

1. Byrd, T., A. Byrd and D. E. Turner, "Measuring the flexibility of information technology infrastructure: exploratory analysis of a construct", *Journal of Management Information Systems*/Summer 2000, Vol. 17, No. 1. pp. 167-208.
2. Casati, F., Castano, S., Fugini, M., Mirbel, I., and Pernici, B. "Using Patterns to Design Rules in Workflows," *IEEE Transactions on Software Engineering* (26:8) 2000, pp 760 - 785.
3. Chau, P. Y.K., K. Y. Tam, "Factors affecting the adoption of open systems: an exploratory study", *MIS Quarterly*, March 1997, pp. 1-24.
4. Cherbakov, L., G. Galambos, R. Harishankar, S. Kalyana, G. Rackham, "Impact of Service Orientation at the Business Level"*, IBM Systems Journal*, Vol. 44, No. 4, 2005.
5. Crowston, K. Crowston and J. Howison, "Hierarchy and Centralization in Free and Open source software team communications", *Knowledge, Technology, & Policy*, Winter 2006, Vol. 18, No. 4. pp. 65-85.
6. Curtis, B., Krasner, H., Iscoe, N., "A Field study of the Software Design Process for Large Systems". *Communications of the ACM*, Nov. 1988, Vol. 31 Issue 11, pp. 1268-1287.
7. Deiters W., Goesmann T., Loffeler T., "Flexibility in workflow management: dimensions and solutions", *Computer Systems Science and Engineering*, 2000, Vol 15; Part 5, pages 303-314.
8. Ducheneanut, N." Socialization in an open source software community: A Socio-Technical analysis", *Computer Supported Cooperative Work*, 2005 14: 323-368.

9. Fitzgerald, B. The transformation of open source software, *MIS Quarterly,* Vol. 30 No 3, pp. 587-598/September 2006.
10. Glynn, E., B. Fitzgerald, C. Exton, "Commercial adoption of open source software: an empirical study", *Proceedings of the 2005 International Symposium on Empirical Software Engineering,* 17-18 Nov. 2005, 10 p.
11. Gordijn, J., Akkermans, H., Van Vliet, H.: Business modeling is not process modeling. *Proceedings of the Conference on Conceptual Modeling for E-Business and the Web,* pp 40-51, 2000.
12. Gulla, J. A., T. Brasethvik, "On the Challenge of Business Modeling in Large-scale Reengineering Projects". *Fourth International Conference on Requirements Engineering*, 2000, 17 p.
13. Hooper, M. J., D. Steeple, C. N. Winters, "Costing customer value: an approach for the agile enterprise", *International Journals of Operations & Production Management*, Vol. 21, No. 5/6, 2001, pp. 630 - 644.
14. Huhns, M. N. and M. P. Singh, "Service-oriented computing: key concepts and principles", *IEEE Internet Computing*, January – February 2005, pp. 75-81.
15. Keen, M., A. Acharya, S. Bishop, A. Hopkins, S. Milinski, C. Nott, Rick R. Robinson, Jonathan J. Adams, Paul P. Verschueren, "Patterns: Implementing an SOA Using an Enterprise Service Bus", *IBM Redbooks*, July 2004.
16. Kraut, R. E.; Streeter, L. "Coordination in Software Development". *Communications of the ACM,* Mar1995, Vol. 38 Issue 3, pp. 69-81.
17. Mcoy, D. W., D. C. Plummer. "Defining, cultivating and measuring enterprise agility", *Gartner Research*, April 28, 2006.
18. Nelson K., Nelson H., Ghods M. "Technology flexibility: Conceptualization, validation, and measurement", *Proceedings of the 30th Hawaii International Conference on System Sciences*, 1997, Volume 3, 10 p.
19. Nelson, K. and Ghods, M. "Measuring technology flexibility", *European Journal of Information Systems*, Volume 7, Number 4, 1 December 1998, pp. 232-240.
20. Ocker, R.J. Fjermestad, J. High versus low performing virtual design teams: a preliminary analysis of communication, *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, 2000**,** 9 p. vol.1.
21. Papazoglou, M. P., D. Georgakopoulos, "Service-oriented computing: introduction", *Communications of the ACM*, 46 (10), (October 2003), 24 – 28.
22. Peltz, C. "Web Services Orchestration: A Review of Emerging Technologies, Tools, and Standards," *Hewlett Packard*, Co., 2003.
23. Petrovic, O., Kittl, C., Teksten, R.D., "Developing Business Models for eBusiness", *the Proceedings of the International Conference on Electronic Commerce 2001*, Vienna, Austria, October 31 - November 4, 2001.
24. Stewart, K. J., S. Gosain, The impact of ideology effectiveness in open source software development teams, *MIS Quarterly,* Vol. 30 No 2, pp. 291-314/June 2006.
25. Stohr, E.A., Zhao, J.L. "Workflow automation: overview and research issues," *Info. Systems Frontiers: Special Issue on Workflow Automation* (3:3) 2001, pp. 281-296.
26. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P. "Workflow patterns," *Distributed and Parallel Databases* (14:3), July 2003, pp. 5-51.
27. Wang, J., and Kumar, A. "A Framework for Document-Driven Workflow Systems," *International Conference Business Process Management*, Nancy, France, 2005, pp. 285-301.
28. Weerawarana, S. *Web services platform architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more*, Prentice Hall PTR, Upper Saddle River, NJ, 2005.
29. Zhao, J. L. "Intelligent Agents for Flexible Workflow Systems", *Proceedings of the AIS Americas Conference on Information Systems,* Baltimore, Maryland, August 14-16, 1998.
30. Zhao, J. L., H. K. Cheng, "Web Services and Process Management: a Union of Convenience or a New Area of Research?" *Decision Support Systems,* 40 (2005), pp.1-8.
31. Zhao, J. L., Tanniru M., Zhang L.-J., "Services computing as the foundation of enterprise agility: overview of recent advances and introduction to the sepcialspecial issue", *Information Systems Frontier,* Volume 9, Number 1 / March, 2007, Pages 1-8.