Ontology-based Workflow Change Management for Flexible eGovernment Service Delivery*

Soon Ae Chun[1,2] and Vijayalakshmi Atluri[1]
[1]MSIS Department and CIMIC, Rutgers University, Newark NJ 07102
[2]MIS Department, Fairleigh Dickinson University, Teaneck NJ 07666
{soon,atluri}@cimic.rutgers.edu

Abstract

Often, eGovernment services needed by citizens are composite in nature, where component services are offered by distributed and autonomous government agencies. Our approach to deliver the inter-agency eGovernment composite services for coordination, cooperation and seamless flow and sharing of data is accomplished through an eGovernment portal system based on a workflow management system (WFMS). Versatile eGovernment service delivery can be achieved through customization of the eGovernment services at the design stage, as well as through run-time adaptation of the services by reacting to changes in the government regulations and mandates, citizens' needs, or unexpected results during service execution. In this paper, we present an approach where the workflows adopt to run-time changes. Given a change condition, our system automatically identifies *migration rules* that specify operations and tasks needed to achieve run-time adaptation and to generate a new workflow. The migration rules are generated from the regulatory rules ontology, which ensure that the new workflow is in some sense consistent with the original workflow.

1 Introduction

Every eGovernment service delivery needs to consider challenges faced by service consumers, i.e., citizens, as well as those faced by service providers, i.e., government agencies. When a traditional interaction of service consumers and providers occurs, citizens interact with one government agency for one particular service at a time. Often the eGovernment services needed by citizens are *composite services* whose component services are offered by distributed and autonomous government agencies who offer their services in conformance with distinct sets of mandates from local, state, and federal governments. A comprehensive digital eGovernment solution requires *mass customization* to consider different qualifications and situations of each citizen and to provide a coherent composite service that is composed of only the applicable component services. Another dilemma is faced by service providers, i.e., individual government agencies is that they do not interoperate with services from other agencies. Since the data are not shared among agencies, it is hard to obtain the overall picture of services consumed by an individual. A successful eGovernment service delivery system requires coordination, cooperation and seamless flow and sharing of data to provide composite services so that the underlying complexity is transparent to the citizens.

Our approach to solve this double challenge in eGovernment service consumption and provision is an eGovernment portal system based on a workflow management system (WFMS). For government service consumers, this system provides a framework to automatically generate *customized workflows* for composite services, which is composed of agency-specific services applicable only to a particular citizen's needs, based on utilizing an eGovernment service ontology and a regulatory rules ontology [3].

In this paper, we propose an approach to handle another challenge: *dynamic changes* in delivering the eGovernment services. Although customized in the initial stage, the workflow in execution may experience changes from various sources, such as the workflow consumer (i.e. citizen) may change preferences, or

*This work is partially supported by the National Science Foundation under grant EIA-9983468, by the NOAA Coastal Services Center under grant NA17OC2587, and by Meadowlands Environmental Research Institute (MERI). We acknowledge Dr. Francisco Artigas, Rutgers and Mia Alpos, MERI for their help in identifying the change types in the land use related regulations and processes.

the external environment may change (i.e. regulations change), or an agency discovered unexpected results that force a change in the given workflow. Given a change condition, our system automatically identifies *migration rules* that specify operations and tasks needed to adjust the workflow to achieve run-time customization. We classify different types of run-time changes. We use a context manager and a change manager module to handle these run-time modifications. The migration process that requires knowledge on a workflow structure, its execution status and the structural manipulations is transparent to the user. This migration knowledge is based on the regulatory rules that govern eGovernment services. Thus, the run-time adjustment obeys the rules and available resource constraints, yielding migration consistency.

Among many techniques and approaches for handling dynamic changes and exceptions in workflow management [4, 1], the approaches in [6, 5] capture exceptions in a knowledge base and provide mechanisms to automatically anticipate, detect, avoid, and resolve them. A rule-based approach [6] determines which running workflow instances and which regions are affected and need adjustment. This approach is different from ours in that it does not utilize the hierarchical conceptual organization of rules. In [5], each specific task is tied to a template with characteristic exception types. Every exception type has an associated knowledge base entry that provides its definition in what situations it is critical, and how it can be handled. This approach is close to our approach in handling the unexpected exception. However, they do not address explicit change requests as in our study.

Types of Dynamic Workflow Changes: Dynamic changes to a workflow may occur due to (1) changes to the goals of the business process (user profile), (2) unanticipated exceptions that arise during a task execution, and (3) changes to the business rules. Consider for example a permit processing workflow: An entrepreneur, Bill, wishes to build a car wash on a 5-acre vacant property that is 400 feet from the bank of the Hackensack River in Little Ferry, New Jersey. Since the zoning for this area is "neighborhood commercial," as designated by the New Jersey Meadowlands Commission (NJMC), the developer needs to obtain Waterfront development and Stream encroachment permits (shown by the solid bubbles in figure 1).
1. Profile Change: While a zoning certificate (task t_2 in figure 1) is under review, assume that Bill submits a request to add a storage mezzanine. This change requires additional parking on the property, and assuming that the property does not have enough space for additional parking, a variance needs to be filed for a public hearing. The stated bulk regulation would interfere with the proposed construction, and pose a hardship, so Bill is asking for an exception to be made. Consequently, the zoning plan review for certificate of approval is suspended, and variance-related tasks t_6 and t_7 need to be inserted, as shown in figure 1.
2. Exceptions: While reviewing the stream encroachment permit application (t_5), suppose that the engineers found some hazardous substances underground. This would require a new task t_9 *notify to remove hazardous materials* before any development (before t_5), as shown in figure 1.
3. Rule Change: Assume that while the development application review is under way, the waterfront regulation that applied to all the developments within 500 feet from the water has changed to apply only to lots within 300 feet distance to obtain a Waterfront permit. Now the Waterfront permit (t_8) needs to be deleted since this rule is no longer applicable, as shown by the dotted line from t_3 to t_5.

2 The Dynamic Workflow Change Management System

Our dynamic change management system, shown in figure 2, has four components: (1) Checker for Change Request Consistency: a mechanism to specify change requests and exceptions, ensuring the request consistency; (2) Context manager (ConMan) that coordinates activities by suspending or resuming the current execution environment via communication with relevant task agents, and provides the current workflow and other required resources for migration; (3) Migration Rule Identification component that identifies migration rules (operations and tasks) requisite for modification, using ontologies of regulatory rules and tasks; and (4) Self Adaptation component that applies migration rules to current workflow. Due to space limitations, in this paper, we present a brief description of our approach, but do not discuss the details of each component. An extended version of this paper is available at [2].

1. Ontology-based Identification of Migration Rules: To identify the necessary migration rules, Self Adaptor first examines the change request type, i.e., user profile, rule, or workflow failure condition. The

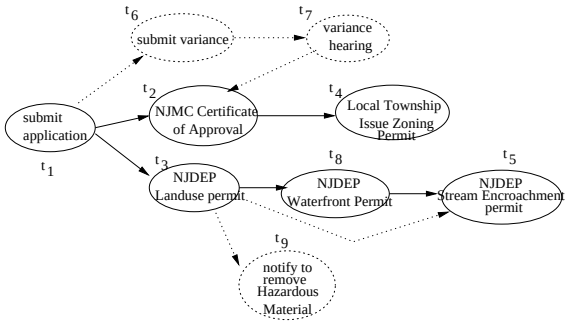

Figure 1: Examples of Workflow Changes

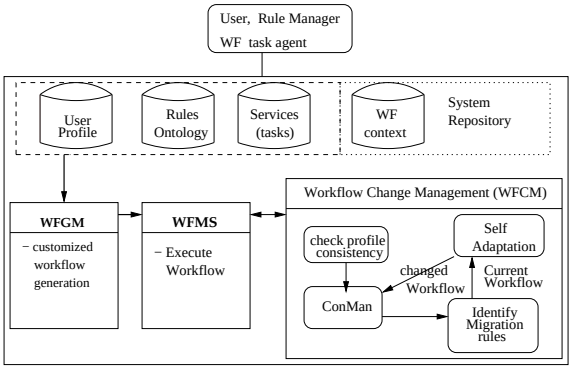

Figure 2: The Dynamic Workflow Change Management System

system searches for rules in the rule ontology that satisfy the change request.

Profile Change and Workflow Condition Change: If the change operation in a change request is 'insert', Self Adaptor identifies a topic node in the rules topic ontology that matches the attribute in the change request. If there is any rule associated with the topic node, the rule is inserted into the set of insert migration rules. Self Adaptor does the same for all the subtopic nodes under this topic node to meet the profile change consistency constraint. When the change operation is 'delete,' Self Adaptor identifies the topic node, as in the case of insert, and applicable rules are added into a set of delete migration rules. However, Self Adaptor checks if the task to be deleted was already executed. If this is the case, it checks whether there is any compensating rule or task, and adds that task to the insert migration rules. When the change operation is 'change,' Self Adaptor needs to identify an existing profile attribute value set to be replaced. It then identifies insertion rules using the new profile values, as in the case of 'insert,' and it identifies deletion rules using old profile values as in the case of 'delete.' Then these sets of insertion and deletion rules are returned as a set of migration rules.

Rule Change: A rule change request has the form of a condition action pair, $\langle c, a \rangle$. If a rule needs to be inserted, Self Adaptor checks if the existing profile satisfies the changed rules condition. If yes, and a is an insertion of a task t, then it checks if t is already in the current workflow. If t exists, then Self Adaptor checks if it was already executed. If yes, then no migration is necessary and the rule is not inserted into the set of migration rules. This avoids redoing t. Otherwise, it inserts the rule into the insertion migration operation. If there is no profile that satisfies this new rule, then there is no need to transform the workflow. The changed rule simply does not apply to the current workflow. If a change request cannot be accommodated, e.g. when a completed task has to be deleted and there is no compensating rule available in the rule ontology, a special flag "point-of-no-return" is returned.

Automatic Exception Handling: To achieve the automatic identification of change needs and to formulate a change request automatically, a WFMS executing t has to recognize the current workflow execution condition as an exception. It is done by identifying any discrepancies between the actual output produced and the *expected output*. When an exception is recognized, the WFMS engine formulates a change request based on the output set and the expected output. The exception handling mechanism uses the *task ontology*. Each task in the task ontology has a set of attributes and relationships. One relationship of a task is *has-exception* that links it to topic nodes in the rules ontology. In other words, each task has relationships to specific exception-related rules topics. Self Adaptor first locates t in the service ontology and follows the link (relationship) *has-exception(t)* to locate exception-related rules. When the exception condition satisfies the exception rule, then the rule is added as migration rule, and the Self adaptation begins. These external exception-related rules are organized in the exception rules topic ontology, which are identified as migration rules, thus handling exceptions in the same way as with the change requests.

2. Applying Migration Rules: Dynamic changes result in restructuring of the workflow, which can either be an insertion or a deletion of a task t_j. Insertions can either be sequential or parallel to a task t_i that has not started its execution yet, or to a task t_k that has completed its execution. In some cases, the changes

require to undo an already completed task and redo it. If it is possible to undo or compensate the completed task, then this change can be accommodated. On the other hand, change requests requiring to undo a completed task that does not have a compensating task will be rejected. Once the migration rules are identified, Self Adaptor applies these migration rules to the current workflow. The Self adaptation process is a dynamic customization of a running workflow to adapt to a changing environment automatically. It considers different types of migration operations, e.g. *compensate, redo, order, insert*, and *delete* in the migration rules identified earlier. Every change is either adapted in the workflow, generating the modified workflow, or a flag to reject the change is returned to ConMan. ConMan sends signals to the appropriate task agents to either resume or cancel current task or restart execution with modified workflows. Self adaptation: Given t_i, the current workflow W, and a set of migration rules, Self Adaptor goes through each migration rule and incorporates a migration operation in the current workflow W. If there is a *point-of-no-return* signal in the migration rule, the change is rejected. First, Self Adaptor looks at all the *compensation* rules in the migration rules, and puts them into a compensation part P_c. Then, it goes through the *redo* rules, and puts them into a redo part P_r. Then, the other migrations are incorporated into W. Each time when a migration rule applies to W, it is updated to W_{new}. If it is an insertion rule (e.g. insert(t)), then Self Adaptor checks if the task to be inserted has any ordering relationships to be met (e.g. order (t, t') or order(t', t)). If there is such a task t', then t will be inserted into W. For a deletion rule, Self Adaptor considers whether it is the first task, last task or a task between two tasks, and incorporates the rule in W. Once W is updated with insertion and deletion migration rules, compensation and redo parts are combined together, P_c followed by P_r, which is attached into the updated current workflow, W_{new}, which is returned to ConMan to resume the execution of the workflow.

3 Conclusions and Future Work

We have provided an ontology-based framework for Workflow Change Management (WFCM) and exception handling for automatic runtime adaptation in response to changes in both the execution environment and user requirements as well as changes in rules and policies that govern the workflow. Our adaptation model allows 1) to express a change request using topic concepts in a rule ontology, obviating the necessity of introducing an artificial change request or manipulation language; 2) to automatically identify migration rules necessary for changes and exception handling with rules ontology, which is different from other studies which assume the user's or a workflow designer's knowledge of the internal structure of the workflow or change knowledge; 3) to automatically apply these migration rules to generate a runtime customized workflow; and 4) to resume and restart the execution.

References

[1] F. Casati et al. Workflow Evolution. *Data & Knowledge Engineering*, 24:211–238, 1998.

[2] S. Chun and V. Atluri. Handling Dynamic Changes in Decentralized Workflow Execution Environments. Technical report cimic-03-1, Rutgers University, 2003. http://cimic.rutgers.edu/ soon/thesis/tech-03.pdf.

[3] S. Chun, V. Atluri, and N. Adam. Domain Knowledge-based Automatic Workflow Generation. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA)*, September 2002.

[4] C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In *Proceedings of Conference on Organizational Computing Systems*, 1995.

[5] M. Klein and C. Dellarocas. A Knowledge-Based Approach to Handling Exceptions in Workflow Systems. *Journal of Computer-Supported Collaborative Work. Special Issue on Adaptive Workf low Systems*, Vol 9(No 3/4), August 2000.

[6] R. Muller and E. Rahm. Rule-Based Dynamic Modification of Workflows in a Medical Domain. In *Proceedings of BTW99*, pages 429–448, Freiburg im Breisgau, March 1999. Springer, Berlin.