응용 프로그램에 독립적인 워크플로우 시스템 모델

최진일° 이상근 최재영
숭실대학교 컴퓨터학부
{jichoi°, sglee, choi}@ss.ssu.ac.kr

Application Independent Workflow System Model

Jinil Choi°, Sangkeon Lee, Jaeyoung Choi
School of Computing, Soongsili University

요 약

워크플로우 시스템은 복잡한 복수의 작업을 사용자가 작성한 작업 플로우에 따라 자동으로 수행시킴으로써 작업의 효율성을 높이고자 하는 시스템이다. 특정 작업의 능률을 향상시키기 위한 목적으로 많은 연구기관에서 자체적인 워크플로우 시스템을 연구하고 있다. 이렇게 개발된 워크플로우 시스템은 각 연구기관에서 목적으로 하는 특정 작업 환경의 특성을 반영하는 시스템이지만 특정 작업이나 환경에 종속적인 특성을 가진다. 따라서 작업 환경이 확장 혹은 변경된다면 워크플로우 시스템도 수정하거나 처음부터 다시 개발해야 한다.

따라서 본 논문에서는 특정 작업이나 환경에 종속적인 워크플로우 시스템의 문제점을 해결하기 위한 방법으로 응용 프로그램 및 실행 환경에 독립적인 워크플로우 시스템 모델을 제안한다.

1. 서 론

워크플로우는 복수의 작업들로 복잡하게 구성되거나 반복적으로 수행되는 작업들을 하나의 작업 흐름으로 기술한 것이다. 워크플로우를 사용하면 사용자가 각각의 작업을 일일이 수행시키지 않고, 하나의 큰 작업으로 수행시킬 수 있어 작업의 능률을 향상시킨다.

최근의 분산 컴퓨팅 환경은 작업을 여러 자원을 활용하여 능률적으로 해결하기 위해 작업을 분할하여 자원별로 할당하고 이를 워크플로우로 통합하여 단일 컴퓨팅 자원으로 해결하기 힘든 문제를 해결하려는 방법들이 활발히 연구되고 있다.

워크플로우는 작업 능률을 최대한 발휘할 수 있도록 여러 응용 프로그램과 이를 실행하는 환경의 특성을 고려하여 작업이 실행되기 전에 정의된다. 이 과정에서 워크플로우 시스템은 특정 응용 프로그램과 실행 환경에 의존적인 특성을 가진다. 때문에 실행되는 환경이 변경된다면 워크플로우와 워크플로우를 수행하는 워크플로우 시스템 역시 수정하거나 다시 개발해야 한다.

그러나 워크플로우의 모든 부분이 실행 환경에 의존적인 것은 아니다. 워크플로우에서 작업들 간의 관계를 나타내는 패턴들[1,2]은 유사하며, 단지 전체적인 패턴들 중에서 해당 워크플로우 환경에 따라 다른 부분 집합이 사용된다고 볼 수 있다. 반면, 워크플로우 패턴들의 연결 대상이 되는 작업들은 워크플로우 환경에 따라 독립적이다. 이러한 문제점들을 해결하기 위해서 워크플로우 패턴들을 통합하고 표준화하기 위한 연구들이 진행되고 있으며, 워크플로우를 구성하는 작업들을 웹서비스를 이용해서 표준화하기 위한 연구 역시 활발히 진행되고 있다.

웹서비스 기술을 적용하여 기존의 워크플로우 작업들을 통합할 경우에, 오버헤드로 인한 성능의 저하와 기존 작업들을 일일이 웹 서비스 인터페이스로 변환하는 작업이 필요한 문제점이 있다.

또, 웹서비스 컨테이너는 웹 서비스 호출을 처리할 뿐, 웹 서비스를 큐잉하거나 스케줄링을 제어할 수 있는 인터페이스를 제공하지 않으므로, 워크플로우를 분산 환경에서 자원별로 분할하여 작업을 수행하기에는 적합하지 않다.

본 논문에서 제안하는 워크플로우 모델은 워크플로우를 구성하는 응용 프로그램의 실행 환경 의존적인 정보를 메타 데이터로 분리하여 관리한다. 이 모델을 사용하면 실행 환경에 의존적인 정보가 직접 워크플로우에 포함되지 않기 때문에, 워크플로우의 재사용성 문제를 해결할 수 있다. 또, 응용 프로그램을 웹서비스로 변환하지 않고 직접 실행하므로 웹서비스를 이용하였을 때보다 성능 상의 오버헤드를 줄일 수 있다.

2. 관련연구

워크플로우를 구성하는 작업들간의 관계를 나타내는 워크플로우 패턴을 표준화하기 위한 연구 사례로는 XPDL, YAWL 등이 있다.

WfMC (Workflow Management Coalition)에서 진행 중인 XPDL (XML Process Definition Language) [3] 프로젝트는 워크플로우를 표준화하여 XML 스키마로 정의하려는 연구이다. XPDL은 작업이 실행되는 다양한 환경의 메타 데이터들을 모아 공통된 부분을 추출하여 작업 흐름간의 인터페이스 연결 부분을 XML 스키마로 정의한

다. 따라서 XPDL을 기반으로 개발된 워크플로우 시스템 간에 연동도 가능하다. 그러나 XPDL은 여러 작업 환경들의 메타 데이터들을 반영하기 위한 것이어서 워크플로우 모델이 매우 복잡하고 방대하므로 구현하기가 어렵다.

워크플로우 패턴에 관한 또 다른 연구 사례로는 YAWL (Yet Another Workflow Language) System[4,5]을 들 수 있다. YAWL System은 Petri Nets을 기반으로 보다 다양한 워크플로우 패턴을 지원할 수 있도록 확장한 것이다. YAWL은 워크플로우 엔진과 작업을 실행하는 브로커를 분리하여 브로커 부분만 구현하면 새로운 형태의 작업을 워크플로우에 통합할 수 있다. 그러나 YAWL System은 웹서비스를 위한 브로커는 지원하지만 기존 응용프로그램을 실행하기 위한 브로커가 지원하지 않는다. 또한 일반 사용자가 사용하기에는 사용법이 쉽지 않다.

분산 컴퓨팅 환경에서 작업을 효과적으로 수행하기 위한 워크플로우로 대표적인 것으로는 DAG_MAN[6]이 있다. DAG_MAN은 분산 컴퓨팅 환경에서 효율적인 워크플로우 관리 기능을 제공하기 위해서 워크플로우에 의존적인 자원들을 스케줄링해주는 기능을 제공한다. 하지만 작업들을 재사용할 수 있는 기능을 제공하지 않는다.

3. 응용 프로그램에 독립적인 워크플로우 시스템 모델

3.1 시스템 구성 요소

본 논문에서 제안하는 워크플로우 시스템 모델은 워크플로우에 직접 작업을 구성하는 응용 프로그램에 관한 내용을 기술하지 않고, 응용 프로그램의 종류나 요구사항만을 기술한다. 응용 프로그램의 입출력, 옵션 등은 태스크 부분에 기술되며, 응용 프로그램의 실행 환경에 의존하는 정보들은 응용 프로그램 풀에 저장되어 워크플로우가 실행될 때 동적으로 작업에 관한 정보가 연결된다. 따라서 워크플로우 시스템은 특정 응용 프로그램이나 이를 실행하는 환경에 대해서 독립적으로 개발할 수 있다.

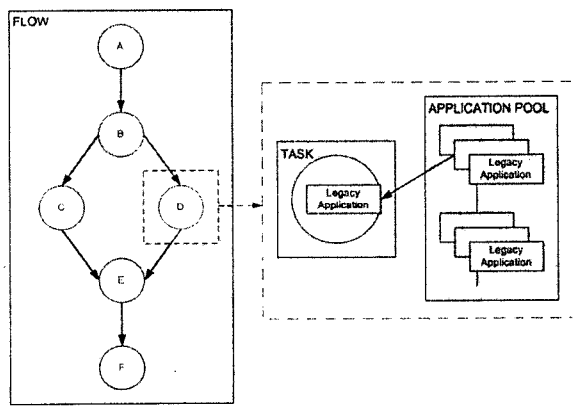

그림 1. 워크플로우 시스템 구성 요소

그림 1은 본 논문에서 제안하는 워크플로우 시스템 모델의 3가지 구성 요소이다. 3가지 구성 요소 중 워크플로우에 관련된 부분은 플로우와 태스크이다.

플로우는 특정 응용프로그램이나 실행 환경과 무관하게 여러 작업간의 진행 그리고 순서와 이들 사이에 발생하는 데이터들의 상호 전달 관계를 표현하기 위한 메타 정보를 가진다. 이러한 메타 정보는 사용자가 쉽게 알아볼 수 있도록 그래프로 표현된다. 그림 1의 플로우에서 원은 태스크에 대한 정보를, 화살표는 작업간의 진행 순서를 표현한다.

태스크는 플로우와 응용 프로그램 사이에서 브로커 역할을 수행하여 실행 환경에 의존적인 응용 프로그램 정보를 워크플로우에 포함시키지 않는다. 때문에, 워크플로우를 응용 프로그램 실행 환경으로부터 독립시킬 수 있다 또한 워크플로우의 재사용성 문제를 해결할 수 있다.

마지막 구성요소로 응용 프로그램 풀이 있다. 응용 프로그램 풀은 워크플로우와 독립적으로 워크플로우 엔진에서 관리되는 요소로 그림 2와 같이 직접적으로 작업을 수행하는 여러 응용 프로그램과 이를 실행하는 환경에 대해 메타 데이터를 가진다. 응용 프로그램 풀은 여러 응용 프로그램들의 정보를 모아 워크플로우 엔진 안에 보관 및 관리한다. 이렇게 관리되는 응용 프로그램들은 사용자 요청에 따라 동적으로 태스크에 할당되어 사용되며, 쉽게 응용 프로그램을 워크플로우 엔진 안에 등록, 그리고 삭제할 수 있다. 이와 같이 워크플로우 시스템은 쉽게 응용 프로그램을 통합하여 확장할 수 있다.

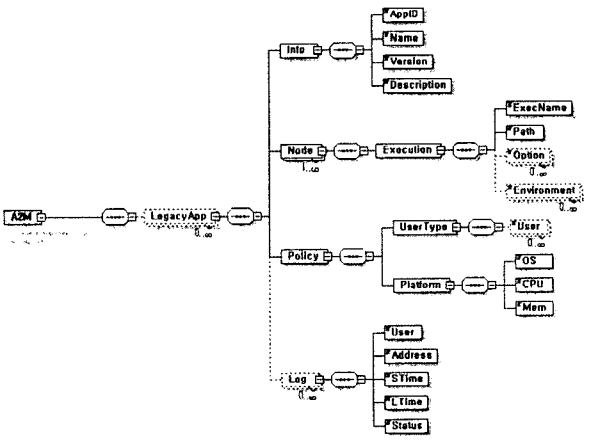

그림 2. 응용 프로그램 풀 XML 스키마

3.2 시스템 구조

본 논문에서 제안하는 응용 프로그램에 독립적인 워크플로우 시스템의 전체적인 구조는 그림 3과 같다. 본 워크플로우 시스템 구조 중에서 사용자와 밀접한 관계를 가지는 부분은 플로우와 태스크가 반영된 편집기 부분이다. 편집기는 사용자와 워크플로우 엔진을 연결하는 부분으로 워크플로우 엔진이 작업을 수행하기 위해 필요로

하는 워크플로우 작성 기능을 제공한다. 또한, 워크플로우 시스템의 확장을 위해 워크플로우와 독립적으로 응용 프로그램을 등록, 수정, 그리고 삭제하는 기능을 제공한다.

워크플로우 엔진은 편집기에서 작성된 워크플로우를 해석하여 작업을 수행하는 부분으로 사용자 관리자, 자원 관리자, 프로그램 관리자, 그리고 실행 관리자로 구성되어 있다. 사용자 관리자는 워크플로우 엔진을 사용할 수 있는 사용자 관리와 편집기를 통해 전달받는 명령 및 워크플로우를 해석하고 처리한다. 자원 관리자는 자원 정보와 사용자 관리자에서 해석된 워크플로우를 바탕으로 태스크 순서를 스케줄링 한다. 프로그램 관리자는 응용 프로그램 풀을 기반으로 편집기를 통해 등록된 여러 응용 프로그램에 대한 정보를 보관 및 관리한다. 또한 다른 관리자들의 요청에 따라 동적으로 응용 프로그램 정보를 제공한다. 실행 관리자는 해석된 워크플로우를 바탕으로 직접 작업을 실행한다.

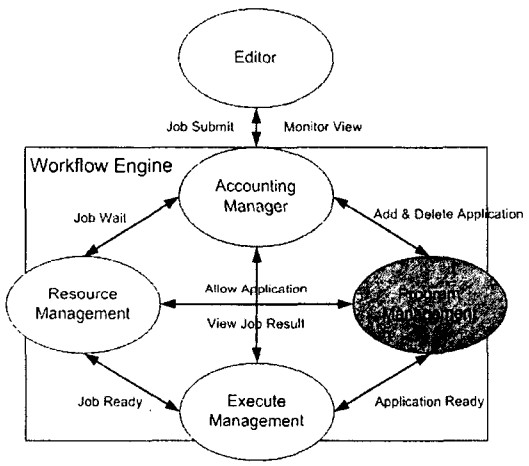

그림 3. 전체적인 시스템 구조

본 논문에서 제안한 워크플로우 시스템의 전체적인 흐름을 살펴보면 우선 사용자가 작업을 수행시키기 위해 워크플로우를 작성한다. 워크플로우를 작성하기 위해서는 작업에 필요한 응용 프로그램 정보를 확인해야 하며, 이를 위해서 편집기를 통해 사용자 관리자에게 응용 프로그램 정보를 요청한다. 요청을 전송받은 사용자 관리자는 프로그램 관리자에게 응용 프로그램 정보를 전송할 것을 명령한다. 명령을 전송받은 프로그램 관리자는 응용 프로그램 풀의 정보를 전송한다. 결과를 전송받은 사용자 관리자는 이를 편집기에 전송한다. 사용자는 응용 프로그램 풀에 대한 정보를 바탕으로 쉽게 워크플로우를 작성할 수 있으며, 작성이 완료된 워크플로우를 사용자 관리자에게 작업을 제출한다. 사용자 관리자는 전송받은 워크플로우를 해석하며, 이에 필요한 자원정보나 응용 프로그램 정보를 해당 관리자를 통해 전달받는다. 이러한 모든 과정이 종료되면 해석된 워크플로우에 따라 태스크를 실행 관리자에게 넘겨준다. 이를 넘겨받은 실행 관리자는 해석된 워크플로우 순서에 따라 태스크를 수행하고 그 결과를 편집기에 전달한다.

4. 결론

기존의 워크플로우 시스템 모델에서 워크플로우는 자체적인 시스템 작업 능률을 최대한 발휘할 수 있도록 특정 응용 프로그램과 이를 실행하는 환경의 특성을 고려하여 사전에 정의된다. 이 과정에서 워크플로우 시스템은 특정 응용 프로그램과 실행 환경에 의존적인 특성을 가진다. 때문에 응용 프로그램이나 이를 실행하는 환경이 변경된다면 워크플로우와 이를 수행하는 워크플로우 시스템 역시 수정하거나 다시 개발해야 한다.

본 논문에서 제안하는 워크플로우 시스템 모델은 응용 프로그램과 이를 실행하는 환경의 정보를 메타 데이터로 분리하여 워크플로우가 아닌 워크플로우 엔진에서 따로 보관 및 관리한다. 따라서 실행 환경에 의존적인 정보가 직접 워크플로우에 포함되지 않기 때문에, 워크플로우의 재사용성 문제를 해결할 수 있으며 응용 프로그램을 쉽게 등록, 수정, 그리고 삭제할 수 있어 워크플로우 시스템의 확장성을 높일 수 있다. 그리고 응용 프로그램을 웹서비스로 변환하지 않고 직접 실행하므로 웹서비스를 이용하였을 때보다 성능 상의 오버헤드를 줄일 수 있다.

향후 연구 방향은 응용 프로그램뿐만 아니라 웹 서비스, 스트리밍 서비스 등 보다 다양한 작업을 지원할 수 있는 워크플로우 시스템을 연구하고자 한다.

5. 참고문헌

[1]. W.M.A. van der Aalst, A.H.M ter Hofstede, B. Kiepuszewski, and A.P. Barros, "Workflow Patterns," BETA Working Paper Series, WP47, Eindhoven University of Technology, Eindhoven, 2000

[2] http://www.workflowpatterns.com

[3] WfMC, "Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface - XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report," Workflow Management Coalition, Lighthouse Point, Florida, USA, 2005.

[4] W.M.P. van der Aalst, "The Application of Petri Nets to Workflow Management," The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[5] W.M.P. van der Aalst and A.H.M. ter Hofstede, "Design and Implementation of the YAWL system," CAiSE 04, Riga, Latvia, June 2004. Springer.

[6] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", presented at 10th International Symposium on High Performance Distributed Computing, 2001.