Microsoft

더욱 강력해진 워크플로우 개발, Windows Workflow Foundation

David Chappell, Chappell & Associates

September 2007

© Copyright Microsoft Corporation 2007. All rights reserved.

Contents

Windows Workflow Foundation 은

사실상 현대 기업에서 사용되는 모든 소프트웨어는 비즈니스 프로세스를 지원한다는 같은 목적을 가지고 있다. 일부 프로세스는 완전히 자동화되어서 애플리케이션 간의 통신에만 의존하지만, 다른 프로세스들은 사람이 프로세스를 시작하고 프로세스가 사용하는 문서를 승인하고, 발생하는 예외사항을 해결한다. 어떤 경우던, 사람과 소프트웨어가 프로세스에 개입하는 액티비티를 의미하는 워크플로우라는, 서로 분리된 일련의 단계들을 지정하는 것이 가능하다. 이 워크플로우가 정의되면, 애플리케이션은 그 정의에 따라 구현되어서 비즈니스 프로세스를 지원하게 된다.

소프트웨어에서 워크플로우를 만들고 실행하는 것은 독특한 어려움을 가지고 있다. 일부 비즈니스 프로세스는 완료되기 까지 몇 시간, 며칠, 몇 주가 걸릴 수도 있다. 이런 긴 시간 동안 워크플로우의 현재 상태에 대한 정보를 애플리케이션이 어떻게 유지해야 하는가? 긴 시간 실행되는 이런 유형의 워크플로우는 다른 소프트웨어와 통신을 하기 마련인데, 개발자는 비동기식 통신 문제를 어떻게 쉽게 해결할 수 있는가? 그리고 소프트웨어 간의 고정된 연동은 모델링하기가 비교적 쉽지만, 사람들은 비즈니스 프로세스를 실행 중에 변경할 수 있는 더 유연한 기능을 원한다. 워크플로우가 이렇게 다양하고 변동이 심한 동작을 어떻게 처리할 수 있을까? 올바른 인프라를 선택하지 않고서는 이런 요구사항을 충족시키기 힘들다. 워크플로우를 지원하도록 분명하게 설계된 기술이 있다면, 이런 유용한 소프트웨어를 만들기 쉬워질 것이다.

마이크로소프트의 Windows Workflow Foundation (WF)은 이런 요구사항을 충족시키기 위해 만들어졌다. .NET Framework 3.0 와 3.5 의 중요한 요소인 WF 는 개발자를 위한 윈도우 플랫폼의 표준 부분이다. WF 는 애플리케이션이 소프트웨어 간의 연동을 조율하고, 사람들 사이의 상호작용을 조율하는, 워크플로우 기반의 애플리케이션을 구현할 수 있는 공용 인프라를 제공한다.

워크플로우 애플리케이션에 필요한 것은

WF 와 같은 워크플로우 애플리케이션에 필요한 것을 이해하려면, 다음의 예와 같이 워크플로우를 사용할 서로 다른 종류의 애플리케이션에 대해 생각해보는 것이 도움이 된다:

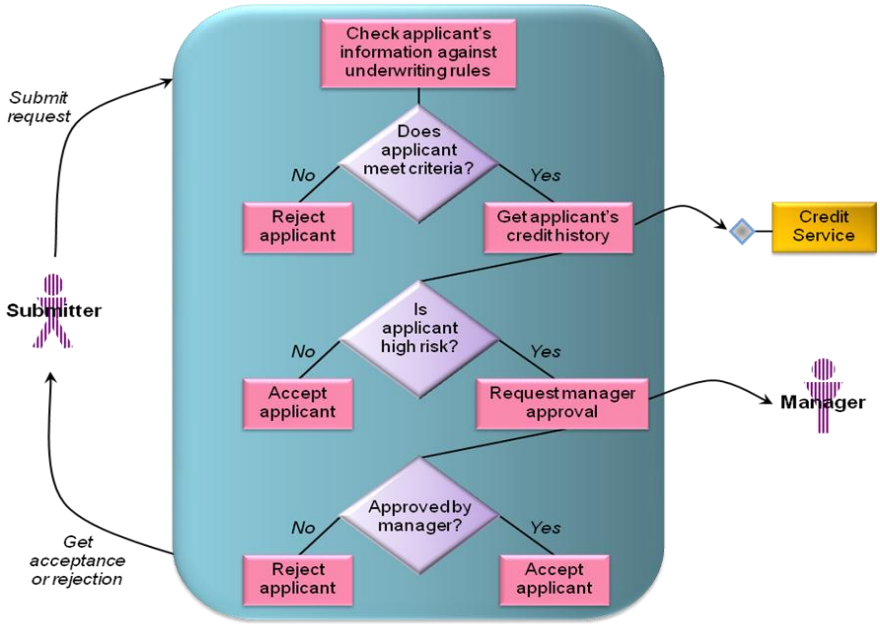
오랜 시간이 걸리는 프로세스를 구현하는 모든 애플리케이션이 워크플로우에 해당된다. 응답하는데 몇 시간 또는 며칠이 걸릴 수 있는 사람들과 상호작용하는 프로세스가 좋은 경우다. 예를 들면, 워크플로우 기반으로 문서 승인 애플리케이션을 구축하는 것이 좋다.

사용자에게 페이지를 보여주는 ASP.NET 애플리케이션은 워크플로우를 사용해서 그 페이지들이 보여지는 순서를 통제한다. 이렇게 하면 페이지 자체를 변경하지 않고도 페이지 흐름을 쉽게 변경할 수 있을 뿐만 아니라 애플리케이션의 사용자 인터페이스를 로직과 분명하게 분리시킬 수 있게 된다.

서비스 지향 환경의 복합적인 애플리케이션은 워크플로우를 사용해서 핵심 동작을 구현할 수 있다. 점점 더 많은 애플리케이션이 서비스를 통해 자신의 로직을 공개하면서, 이런 서비스를 활용해 비즈니스 프로세스를 만드는 것이 더 쉬워지고 있다. WF 와 같은 워크플로우 기술은 서비스들을 호출해서 하나의 복합 애플리케이션으로 만들어나가는 로직의 인프라를 제공한다.

CRM 과 같이 특정 문제를 또는 재무회계 서비스와 같이 특정 수직 시장을 대상으로 하는 경우 워크플로우 기반이 좋다. 이런 애플리케이션은 일반적으로 한 두 가지만 다른 비즈니스 프로세스를 구현한다. WF 와 같은 공용 인프라에서 이런 프로세스를 추진하는 로직을 구현하게 되면 애플리케이션을 더 쉽게 구축하고, 더 쉽게 변경하고 사용자 지정할 수 있게 된다.

워크플로우 프레임워크가 해결해야 하는 요구사항을 알 수 있는 가장 좋은 방법은 예제 워크플로우 애플리케이션을 좀 더 자세하게 살펴보는 것이다. ISV 가 보험 회사를 위한 워크플로우 애플리케이션을 만들고 싶어한다고 가정해보자. 다음의 그림은 자동차 보험 증서를 요청하는 애플리케이션의 간단한 예를 보여주고 있다.



제출자가 새 증서 요청을 전송하면서 프로세스가 시작된다. 이 제출자는 콜센터의 직원, 현장의 보험 대리인 또는 인터넷을 통해 요청하는 고객 중 한 사람이 된다. 새 요청이 도착하면 이 워크플로우의 새 인스턴스가 만들어진다. 워크플로우는 요청 정보를 회사의 증서 발행 규칙과 비교확인하면서 시작된다. 신청자가 회사의 기준에 못 미친다면, 신청은 거절된다. 기준에 도달한다면, 워크플로우는 외부 신용 서비스에게 신청자의 신용 경력을 요청한다. 만족스러운 신용점수는 즉시 승인이 되지만, 신용점수가 나쁘고 높은 위험의 신청자는 관리자의 승인이 필요하게 된다. 관리자가 승인하면 신청이 받아들여지지만 그렇지 않다면 신청이 거부된다.

이 간단한 예는 일반적인 워크플로우 기반의 애플리케이션이 요구하는 많은 것을 보여주고 있다. 요구사항으로는 다음과 같은 것들이 있다:

비즈니스 규칙에 따라 의사결정을 내릴 수 있는 능력. 신용 확인 결과에 따른 예 또는 아니오의 의사결정과 같은 간단한 규칙과 초기 의사결정을 내리기 전에 반드시 평가해야 하는 크고 복잡한 규칙 등이 있다.

워크플로우 외부의 다른 소프트웨어와 시스템과 통신할 수 있는 방법. 이 예에서는, 초기 요청은 ASP.NET 페이지와 같은 애플리케이션의 다른 부분에서 수용되는 반면에, 신용 서비스 접속과 같은 부분은 웹 서비스 또는 다른 기술을 사용해 통신해야 한다.

사람과 상호작용 할 수 있는 방법. 여기에서는, 관리자가 일부 신청자를 승인해야 하기 때문에, 워크플로우는 사용자 인터페이스 자체를 보여주거나 다른 소프트웨어를 통해 사람과 상호작용할 수 있어야 한다.

워크플로우 수명의 전과정에서 상태를 유지할 수 있는 능력. 특히 이 예의 관리자에서와 같이 사람이 개입될 경우, 워크플로우가 완료되려면 오랜 시간이 걸릴 수 있다. 관리자가 며칠 동안 자리를 비우거나 휴가를 냈다면 어떻게 될까? 확장성 높은 시스템을 구축하려면 워크플로우를 해제시키고 그 상태를 저장한 후에 다음 단계를 실행할 수 있는 경우에 다시 활성화시키고 그 상태를 로딩할 수 있어야 한다.

이 모든 요구사항은 워크플로우의 기본 요구사항이다. 대부분의 요구사항은 워크플로우를 명시적으로 지원하도록 설계된 인프라가 없다면 해결하기 힘든 것들이다.

워크플로우 프레임워크는 이런 기본 기능 이상을 제공한다. 예를 들면, 다음과 같은 기능을 제공한다:

각 단계가 소프트웨어의 특정 부분으로 구현될 수 있는, 컴포넌트 비슷한 방식. 보험 애플리케이션 또는 시스템 관리와 같은, 특정 영역에 도움이 되는 미리 정의된 단계를 만든 후에, 다양한 워크플로우에서 이 단계들을 사용할 수 있다.

워크플로우를 그래픽으로 만들고 변경할 수 있는 툴. 한 워크플로우는 정의된 단계들로 구성되기 때문에, 이 단계들과 그 관계를 눈으로 보여주는 툴을 사용해서 워크플로우를 만들 수 있다.

실시간으로 워크플로우의 실행을 검사하면서 모니터링할 수 있는 능력.

단계 하나를 추가해서 실행되고 있는 워크플로우 인스턴스를 변경할 수 있는 방법. 소프트웨어만 사용될 경우에는 특별하게 필요한 능력이 아니지만, 사람과 상호작용해야 할 때에는 이런 유연성이 반드시 필요하다. 워크플로우는 각각이 고유의 장점과 요구사항이 있는 독특한 기술이다. 이 방식이 전문화된 영역에 오랜 동안 적용되어왔지만, 워크플로우를 만들 수 있는 범용적인 인프라를 통해 더 널리 사용될 수 있게 되었다. 워크플로우 기술이 소프트웨어 개발의 주류가 될 것으로 기대를 모았지만 현실은 그렇지 않았다. 이제 워크플로우가 새롭게 관심을 모으기 시작했다.

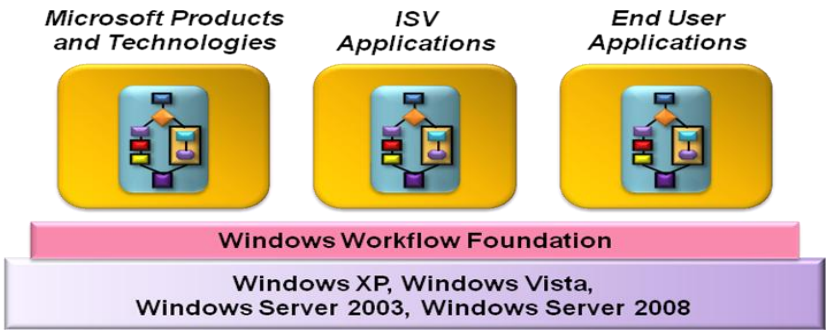
Windows Workflow Foundation 이 제공하는 것

워크플로우를 위한 범용적인 인프라인, WF 는 다양한 목표를 가지고 있다. 그 중에서도 윈도우용 공용 워크플로우 기술을 제공하고, 다양한 워크플로우 애플리케이션의 인프라를 제공하고, 시스템과 사람의 워크플로우를 통합한다는 세 가지 목표가 가장 중요하다. 여기에서는 이 각각의 목표를 설명하고 있다.

윈도우용 워크플로우 기술을 제공

많은 윈도우 애플리케이션은 어느 정도의 워크플로우 지원기능을 가지고 있다. 마이크로소프트 제품만을 생각하면, BizTalk Server 와 Exchange Server 등의 워크플로우 기술을 사용할 수 있다. 이런 광범위한 사용은 워크플로우가 얼마나 유용한지를 잘 알려주지만, 윈도우용으로 다양한 워크플로우 기술을 가지고 있을 필요는 없다. 다른 주류 개발 기술과 마찬가지로, 워크플로우가 윈도우 플랫폼 자체의 기본 기능으로 지원되고 있기 때문이다.

이 기본 기능은 WF 가 제공하는 것이다. 워크플로우를 .NET Framework 의 일부로 구현하고 있기 때문에, 워크플로우가 필요한 모든 윈도우 애플리케이션에서 사용할 수 있다. 다음 그림은 이런 개념을 잘 보여주고 있다.

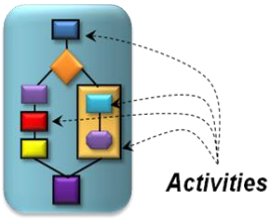


그림에서와 같이, WF 를 클라이언트와 서버 컴퓨터 모두에서 사용할 수 있으며 일반 사용자, ISV, 마이크로소프트가 만든 애플리케이션에서 사용할 수 있다. 약간의 시간이 더 필요하겠지만, WF 는 마이크로소프트 제품과 기술의 워크플로우 공용 인프라가 될 것이다. WF 기반으로 문서 지향의 워크플로우 서비스를 제공하는, Windows SharePoint Services 가 가장 좋은 예다. BizTalk Server 와 같은 다른 마이크로소프트 제품의 다음 버전도 WF 를 사용해서 워크플로우 서비스를 구현할 것이다.

WF 는 일반 사용자가 바로 사용할 수 있는 워크플로우 애플리케이션이 아니라, 개발자를 위한 플랫폼이라는 것을 이해하고 있어야 한다. 그렇기 때문에, 정보 근로자가 워크플로우를 사용할 수 있게 하는 툴을 제공하지 않는다. 또한 WF 는 툴을 만들 수 있는 정보를 제공하고 있지만, 완전한 기능의 관리와 모니터링 툴을 제공하지 않는다. WF 의 목적은 완벽한 윈도우용 워크플로우 솔루션이 아니라, 소프트웨어 개발자가 워크플로우 기반의 윈도우 애플리케이션을 쉽게 만들 수 있게 하는 것이다.

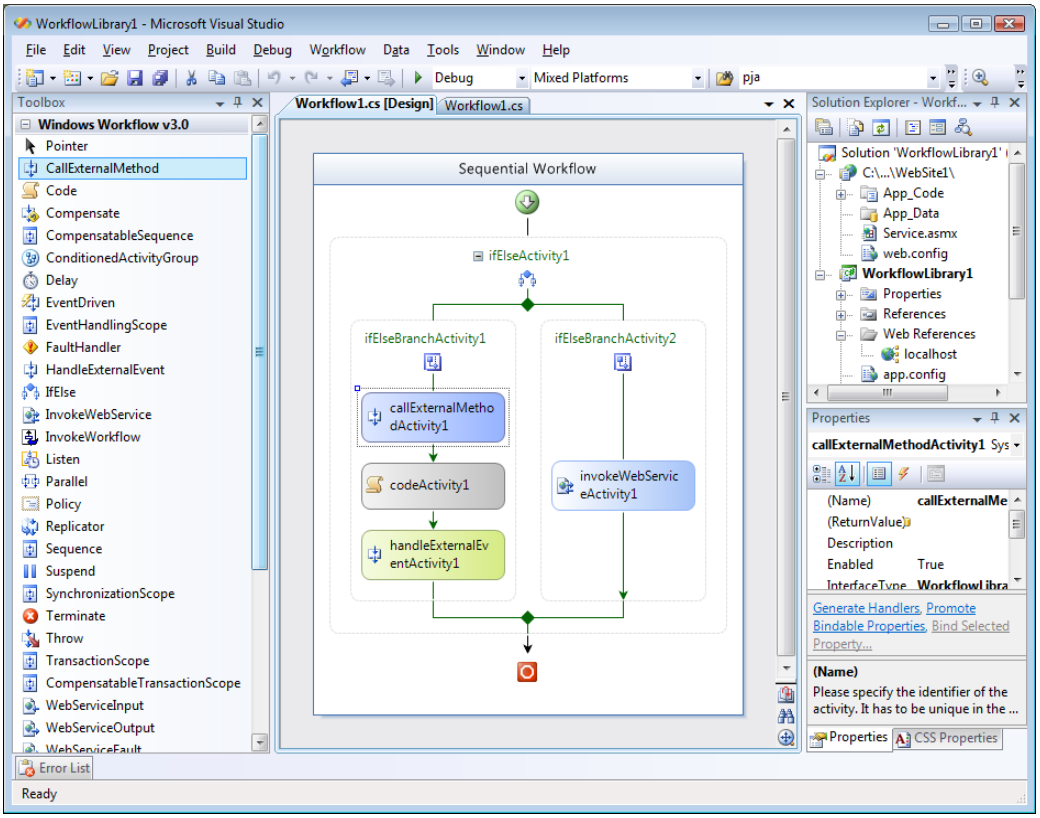
다양한 워크플로우 애플리케이션을 위한 인프라 제공

윈도우용 공용 워크플로우 기술 개발은 상당히 어려운 문제를 가지고 있다. 윈도우 애플리케이션이 워크플로우 기술을 다양한 방식으로 사용한다는 것을 감안하면, 어느 한 솔루션이 이 모든 요구사항을 성공적으로 해결할 수 있겠는가? 워크플로우를 표현하는 한 언어와 워크플로우를 정의하는 한 그래픽 툴을 구현하는 전통적인 워크플로우 제품으로는 이렇게 다양한 요구를 충족시킬 수 없다. 다른 방식이 필요하다. WF 는 한 언어와 한 툴을 제공하지 않고, 워크플로우를 만들고 실행할 수 있는 범용 워크플로우를 제공한다. WF 에서는, 워크플로우는 다음 그림과 같이 액티비티 (activities) 그룹으로 구성되며, 이 액티비티들은 워크플로우의 특정 동작을 실행한다.



WF 는 워크플로우를 정의하기 위한 다목적 액티비티들이 함께 제공되고 있다. 베이스 액티비티 라이브러리 (base activity library)는 if/else 와 while 순환문과 같은 친숙한 구문을 사용해서 흐름 통제를 정의할 수 있는 능력을 제공한다. 또한 이 라이브러리는 Windows Communication Foundation (WCF)를 사용해서 다른 소프트웨어와의 통신을 지원하는, 규칙 엔진이 포함되어 있다. 그렇지만 이런 내장 액티비티들이 상당히 많은 워크플로우 기능을 제공하지만, 개발자들도 특정 문제 영역에 집중하는 사용자 지정 액티비티를 자유롭게 구현할 수 있다. 실제로, 액티비티는 워크플로우의 컴포넌트 모델과 매우 흡사한 기능을 제공한다. 예를 들면, Windows SharePoint Services 는 문서 지향적인 워크플로우를 만들 수 있는 액티비티들을 제공한다. 다른 애플리케이션들은 자신만의 액티비티 그룹을 자유롭게 만들 수 있다. Windows SharePoint Services 에서는, 워크플로우가 사용자에게 과업을 할당할 수 있도록 CreateTask 와 CompleteTask 등의 액티비티를 제공한다. 워크플로우 기반의 의료 서비스 애플리케이션은 완전히 다른 사용자 지정 액티비티들을 제공하지만, 워크플로우 기반의 보험 애플리케이션은 다른 액티비티들을 정의할 수 있다. 사용자 지정 액티비티는 WF 와 함께 제공되는 액티비티와 같은 공용 인터페이스를 사용하기 때문에, 워크플로우는 사용자 지정 액티비티와 베이스 액티비티 라이브러리의 액티비티를 결합할 수 있다. 사용자 지정 액티비티 개발자들도 WF 의 베이스 액티비티 라이브러리를 사용하지 않아도 된다. 능력있는 고급 개발자들이 만든 도메인 고유의 액티비티를 훨씬 많은 개발자들이 활용해서 워크플로우를 만들 수 있다.

WF 워크플로우는 코드로 직접 작성할 수 있다. 워크플로우는 그래픽으로 정의한 후에 필요한 경우에 코드를 추가할 수 있다. 이것이 가능하도록, WF 는 개발자가 워크플로우를 만들고 변경할 수 있는 Workflow Designer 를 제공한다. 다음 그림과 같이, Workflow Designer 는 Visual Studio 내에서 실행될 수 있다. 왼쪽의 툴박스는 베이스 액티비티 라이브러리 아이콘을 가지고 있기 때문에, 이것을 디자인 공간에 드랙앤드롭해서 워크플로우를 만들 수 있다. 각각의 액티비티는 Visual Studio 에서 개발자가 설정할 수 있는 속성과 이벤트를 가지고 있다.



또한 Workflow Designer 는 다른 개발환경에서도 실행되고 사용자 지정될 수 있다. 워크플로우를 자신의 제품에 포함시키고 싶은 ISV 는 이 툴을 제품 안에서 직접 실행해서 원하는 룩앤필을 제공할 수 있다. 이와 비슷하게, 정보 근로자들이 워크플로우를 만들고 변경할 수 있는 방법을 제공하려는 기업은 Visual Studio 대신에 더 친숙한 환경 내에서 Workflow Designer 를 실행할 수 있다. 그리고 ISV 는 Workflow Designer 를 사용하지 않고 다른 그래픽 디자인 툴을 사용해서 WF 워크플로우를 만들 수 있다.

윈도우용 범용 워크플로우 인프라를 제공하는데 있어서 또 다른 문제점은 호스팅이다. 어떤 종류의 애플리케이션이 워크플로우를 실행할 수 있어야 할까? WF 의 답변은 워크플로우가 거의 모든 윈도우 프로세스에서 실행될 수 있게 하는 것으로, 간단한 콘솔 또는 Windows Forms 애플리케이션에서 기업 또는 ISV 가 제공하는 훨씬 복잡한 서버 소프트웨어 모두에서 실행될 수 있다. 이것은 WF 가 광범위한 소프트웨어에서 사용될 수 있게 하려는 것이다.

WF 개발자의 주요 목적은 이 워크플로우 프레임워크를 활용하는 소프트웨어 생태계를 지원하려는 것이다. 독특한 문제 도메인, Workflow Designer 를 호스팅하는 사용자 지정 환경 그리고 WF 런타임을 위한 독특한 호스팅 환경을 위한 특수 액티비티를 만들어서 특수한 요구사항들을 해결할 수 있다. 그렇지만 WF 가 사용되는 방식과 상관없이, 핵심 프로그래밍 모델과 런타임 엔진은 같은 것이며, 그래픽 개발 툴의 동작도 비슷할 것이다. WF 사용자 경험은 시간이 지나면서 상당히 달라지겠지만, 사용자가 경험하는 인프라는 계속 공용 상태로 남게 될 것이다.

시스템과 사람의 워크플로우 통합

비즈니스 프로세스는 일반적으로 사람과 애플리케이션 모두가 필요하지만, 소프트웨어 간의 연동을 자동화하는 것과 사람 간의 상호작용을 자동화하는 것은 완전히 다르다. 그렇기 때문에, 워크플로우 기술은 일반적으로 어느 한쪽만을 강조해왔다. WF 의 주요 목적 중 하나는 두 문제를 해결하는 통합 솔루션을 제공해서 이런 편중을 끝내는 것이다.

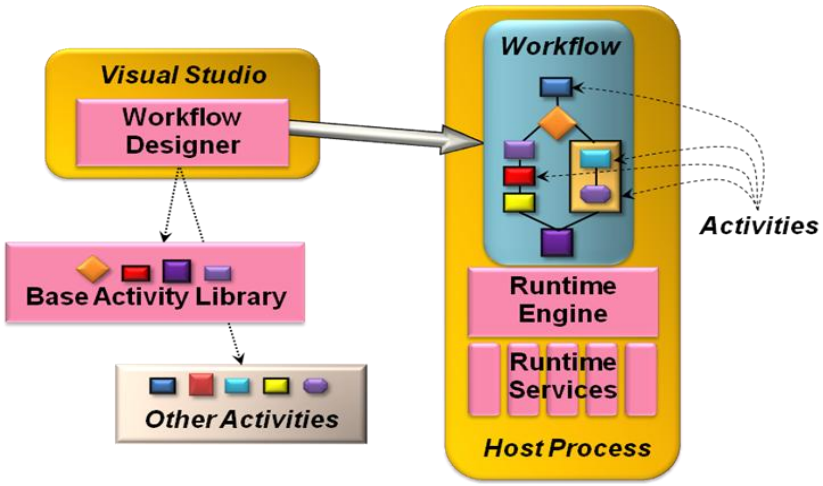
이것이 왜 문제가 되는지 이해하기 위해, 애플리케이션 간의 연동 자동화가 가지는 전형적인 특징에 대해 생각해보자. 시스템 워크플로우 또는 오케스트레이션으로 알려진, 애플리케이션 연동은 예측이 가능하며 비교적 정적이다. 이 연동을 주도하는 로직은 한 번만 정의된 후에 변경하지 않고 계속해서 사용할 수 있다. 또한 시스템 워크플로우는 XML 문서와 같이 구조적이고, 잘 정의된 데이터를 교환하기 때문에 사람의 개입없이도 소프트웨어가 효과적으로 처리할 수 있다.

오케스트레이션을 사람 간의 상호작용을 조율하는 휴먼 워크플로우와 비교해보자. 사람을 대신해서 동작하는 애플리케이션은 다른 소프트웨어와만 연동되는 애플리케이션보다 훨씬 더 유연해야 한다. 사람들은 마음을 바꾸고, 새로운 생각과 예외를 개입시키고, 갑자기 프로세스를 취소하고 휴먼 워크플로우를 다이나믹하게 만드는 일들을 한다. 그렇기 때문에 휴먼 워크플로우를 지원하는 소프트웨어는 더 이벤트 중심이고 유연한 방식을 허용해야 한다. 휴먼 워크플로우는 일반적으로 이메일과 텍스트 문서와 같이 반구조적이고 사람이 읽을 수 있는 데이터를 사용한다.

이 두 워크플로우의 서로 다른 요구사항을 감안하면, 한 기술로 두 요구사항을 모두 해결하는 것이 당연히 어려울 수 밖에 없다. 그렇지만 WF 는 통합 방식으로 두 요구사항을 해결하는데 초점을 맞추고 있다. WF 는 워크플로우를 순차적, 이벤트 기반으로 정의하고, 실행되고 있는 워크플로우에 단계를 추가하거나 변경할 수 있는 능력을 제공한다.

Windows Workflow Foundation 사용하기

WF 가 제공하는 기능은 쉽게 이해할 수 있다. 그렇지만 이 기술이 해결하는 것을 진정으로 이해하려면 더 많은 것을 살펴볼 필요가 있다. 여기에서는 WF 의 다양한 부분을 설명하고 있다. 먼저, 다음 그림과 같이 전체 그림을 살펴보도록 하자.



이 다이어그램이 보여주고 있듯이, WF 의 기본 구성요소는 다음과 같다:

액티비티 (Activity): 작업 단위. 액티비티가 구현하는 작업은 매우 간단하거나 상당히 복잡할 수 있다.

워크플로우: 어떤 프로세스 로직을 구현하는 액티비티 그룹. ₩Workflow Designer: Visual Studio 에서 실행되어 WF 워크플로우와 액티비티를 만들고 변경할 수 있는 그래픽을 베이스 액티비티 라이브러리 (Base activity library): 개발자가 사용해서 워크플로우를 만들 수 있는 베이스 액티비티 그룹. 워크플로우는 다른 액티비티를 자유롭게 사용할 수 있다

런타임 엔진: 워크플로우를 실행하는 WF 라이브러리. 런타임 엔진은 워크플로우 외부의 소프트웨어와 통신하는 메커니즘과 같은 다른 서비스를 제공한다.

런타임 서비스: 트랜잭션 지원, 워크플로우 상태 저장, 워크플로우 실행 추적과 같이, 워크플로우가 사용할 수 있는 다양한 서비스들.

호스트 프로세스: WF 런타임 엔진과 이것이 실행하는 모든 워크플로우를 호스팅하는 윈도우 애플리케이션.

WF 를 이해하려면 이 모든 구성요소에 대해 알아야 한다. 먼저 다른 모든 것이 존재하는 이유인 워크플로우에 대해 알아보도록 한다.

워크플로우에 대한 이해

모든 WF 워크플로우는 어느 정도의 액티비티를 가지고 있으며, 이 액티비티는 그 워크플로우의 특정 기능을 수행한다. 워크플로우는 이런 액티비티들의 컨테이너 역할을 해서, 액티비티의 주기와 실행 순서를 통제할 수 있는 방법을 제공한다.

WF 는 통합 방식으로 시스템과 휴먼 워크플로우 모두를 지원하려고 하는데, 이것 때문에 문제가 발생될 수 있다. 앞에서 설명한 것과 같이, 시스템 워크플로우는 잘 정의되고, 예측가능한 방식으로 액티비티를 실행하는 경향이 있지만, 휴먼 워크플로우는 그렇지 않다.

두 워크플로우의 요구사항을 모두 해결하기 위해, WF 는 미리 정의된 패턴으로 액티비티를 실행할 수 있는 순차 워크플로우 (sequential workflows) 형식과 외부 이벤트가 발생되는 것에 대응할 수 있는 상태 시스템 워크플로우 (state machine workflows) 형식을 제공한다. 두 형식 모두 같은 런타임 환경을 사용하며, 같은 사용자 지정 액티비티를 사용한다. 순차 방식은 시스템 워크플로우에 적합한 반면에, 상태 시스템은 휴먼 워크플로우의 더 느슨하게 정의된 성격을 모델링할 수 있는 방법을 제공한다. 한 워크플로우는 두 형식의 요소를 결합할 수 있기 때문에 두 형식이 결합될 수 있다.

필요하다면, 개발자는 사용자 지정 워크플로우 형식을 만들 수도 있지만, WF 애플리케이션은 두 표준 옵션 중 하나를 사용하는 경우가 가장 많다. 다음은 순차 워크플로우에 대해 설명하고 있다.

순차 워크플로우 사용하기

순차 워크플로우는 워크플로우 액티비티가 잘 정의된 순서로 실행되는 애플리케이션을 위한 것이다. 이 순서는 순환과 분기 등의 흐름 통제 기능을 가지고 있으며 워크플로우는 시작에서 종료까지 정의된 경로를 가진다. WF 와 함께 제공되는 기본 액티비티 라이브러리는 순차 워크플로우에서 사용될 수 있는, 다음과 같은 액티비티 그룹을 가지고 있다:

IfElse: 조건이 충족되었는 지에 따라 한 개 이상의 가능한 경로에 있는 액티비티를 실행한다.

While: 조건이 참인 동안에는 한 개 이상의 액티비티를 반복적으로 실행한다.

Sequence: 정의된 순서로 한 번에 한 액티비티 그룹을 실행한다.

Parallel: 두 개 이상의 액티비티 순서를 실행하지만, 모든 순서가 완료되어야 계속 실행된다.

Code: 정의된 코드 부분을 실행한다.

CompensationHandler: 다른 액티비티 그룹이 실행 도중에 에러가 발생될 경우, 실행되는 대리 로직과 코드를 가지고 있다.

Listen: 특정 이벤트를 대기한 후에, 그 이벤트가 수신되면 한 개 이상의 액티비티를 실행한다.

Delay: 지정된 시간 동안 워크플로우의 실행을 중단한다.

CallExternalMethod: 이 애플리케이션에 있지만 워크플로우 외부에 있는 객체의 메서드를 호출한다.

HandleExternalEvent: 이 애플리케이션에 있지만 워크플로우 외부에 있는 또 다른 메서드의 호출을 기다린다.

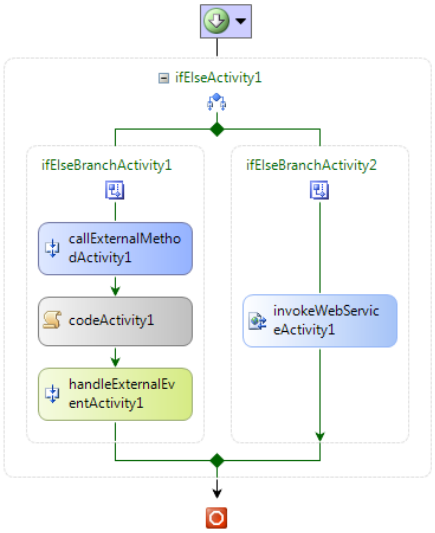
InvokeWorkflow: 또 다른 워크플로우가 실행을 시작하게 만든다.

InvokeWebService: 웹 서비스를 호출한다.

TransactionScope: 한 개 이상의 액티비티가 완료한 작업을 하나의 원자 (atomic) 트랜잭션으로 결합할 수 있게 한다.

Terminate: 워크플로우의 실행을 종료한다.

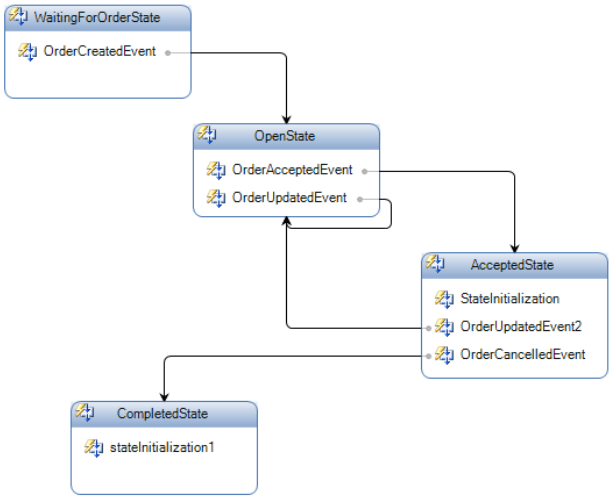
다음의 다이어그램은 WF Workflow Designer 와 여러 액티비티를 사용해서 만들어진 간단한 순차 워크플로우를 보여주고 있다.



이 분야에 약간의 경험을 가진 사람은 순차 워크플로우에 사용된 기본 액티비티 중 많은 것이 Business Process Execution Language (BPEL)의 액티비티와 비슷하다는 것을 알았을 것이다. 처음에는 마이크로소프트와 IBM 이 정의했던 BPEL 은 이제 OASIS 표준이 되었다. 이 언어는 시스템 워크플로우를 정의하기 위한 것으로 WF 에서 더 범용적인 방식으로 지원하고 있다. BPEL 을 사용하려는 개발자들을 위해, 마이크로소프트는 BPEL for Windows Workflow Foundation 이라는 기술을 제공하고 있다. 이 옵션을 사용해서, BPEL 에서 정의된 프로세스 로직을 WF 워크플로우에 가져올 수 있다. 또한 개발자들은 WF 워크플로우 로직을 BPEL 로 내보낼 수 있으며 BPEL 기반의 WF 액티비티를 직접 사용할 수 있다.

상태 시스템 워크플로우 사용하기

액티비티를 미리 정의된 패턴으로 만드는 순차 워크플로우와 달리, 상태 시스템 워크플로우는 액티비티를 한정된 상태 시스템으로 조직한다. 아래의 그림은 WF Workflow Designer 을 사용해 만든 간단한 상태 시스템 워크플로우를 보여주고 있다. 이 그림이 보여주듯이, 개발자는 상태와 이벤트 그룹을 정의하고 있으며, 이 정의에 따라 상태 간의 전환을 유발 (Trigger)시킨다.



워크플로우 액티비티를 이렇게 조직하면, 이벤트 중심의 비즈니스 프로세스와 같이 이벤트의 정확한 순서를 미리 알지 못하거나, 상당히 많은 가능성으로 가능한 모든 경로를 정의하지 못할 경우에 많은 도움이 된다. 사람의 작업을 조율하는 워크플로우가 좋은 예가 될 수 있다. 상태 시스템 워크플로우는 실행 중에 상태를 쉽게 취소할 수 있고 (좋은 고객은 신용 확인을 생략), 비즈니스 프로세스의 어떤 단계로도 건너 뛰거나 (높은 우선 순위를 바로 처리) 언제라도

비즈니스 프로세스를 취소할 수 있다 (고객이 주문을 취소). 상태 시스템은 기업이 생각하고 비즈니스 프로세스를 문서로 만드는 워크플로우를 정의할 수 있는 방법도 제공한다. 이런 경우, 이런 식으로 워크플로우를 모델링하게 되면 개발자와 비즈니스 인력이 보다 효과적으로 협업할 수 있게 된다.

베이스 액티비티 라이브러리는 상태 시스템 워크플로우를 만들 수 있도록 명시적으로 설계된 다음과 같은 액티비티가 있다:

State: 한 워크플로우의 상태 시스템에서의 상태.

EventDriven: 특정 상태인 동안에 특정 이벤트가 수신되면 실행되어야 할 한 개 이상의 액티비티를 가지고 있는 전환 (transition)을 정의한다.

SetState: 워크플로우의 상태 시스템 상태를 변경한다. 전환이 워크플로우의 상태를 변경하거나 변경할 수 없다.

StateInitialization: 특정 상태가 될 때 마다 기본적으로 실행되어야 하는 한 개 이상의 액티비티를 정의한다.

StateFinalization: 특정 상태를 빠져 나오면 기본적으로 실행되어야 할 한 개 이상의 액티비티를 정의한다.

위의 예에서는 나타나지 않았지만, 상태를 중첩 (nest)시킬 수도 있다. 외부 상태에 정의된 이벤트는 그 상태와, 그 상태가 가지고 있는 모든 상태에 적용된다. 이렇게 되면 주문 취소와 같이, 이런 상태들 중 어느 하나에 언제라도 발생될 수 있는 이벤트를 쉽게 표현할 수 있다.

상태 시스템 워크플로우는 순차 워크플로우를 위한 액티비티들도 사용할 수 있기 때문에, 각각의 전환에서 취한 동작들이 액티비티 순서와 다른 로직을 가지고 있을 수 있다. WF 는 기본 워크플로우 스타일을 결합해서 사람과 시스템 워크플로우에 필요한 서로 다른 형식들을 통합 방식으로 지원한다.

워크플로우 만들고 변경하기

WF 워크플로우를 만들고 변경할 수 있는 가장 쉬운 방법은 Workflow Designer 를 사용하는 것이다. 앞에서 설명한 것과 같이, Designer 의 기본 호스트는 Visual Studio 로, 여기에서 순차 워크플로우, 상태 시스템 워크플로우 등을 만들 수 있는 프로젝트 템플리트를 추가한다. 개발자는 다양한 액티비티를 디자인 면에 드랙앤드롭하고 속성을 설정해서, 새 워크플로우를 만들고 기존 워크플로우를 수정할 수 있다. 이 방식은 개발자가 컨트롤을 양식에 드롭해서 GUI 를 만드는 Windows Forms 또는 Windows Presentation Foundation 과 비슷하다. Workflow Designer 를 사용해서 워크플로우에 액티비티를 드롭하면 비즈니스 로직이 만들어진다. 두 방식 모두 개발자의 생산성을 크게 높여준다.

Workflow Designer 는 C#과 Visual Basic 만을 지원한다. WF 의 서비스를 사용하고 싶지만 코드에서 자유롭게 사용하고 싶은 개발자는 Workflow Designer 를 사용하지 않아도 된다. 이 경우, 개발자는 C#, Visual Basic, C++ 등, .NET Framework 의 Common Language Specification 와 호환되는 모든 개발언어를 사용할 수 있다.

한 꺼풀 벗겨내면, 워크플로우는 클래스일 뿐이다. System.Workflow.Activities, System.Workflow.ComponentModel, System.Workflow.Runtime 의 세 네임스페이스는 워크플로우와 관련 액티비티들을 만들고 실행하는 데 필요한 형을 제공하고 있다. 예를 들어, 새 순차 워크플로우를 만들기 위해, C# 개발자는 다음과 같은 코드 뼈대를 사용할 수 있다:

```
using System.Workflow.Activities;
public class ExampleWorkflow : SequentialWorkflow
{
    …
}
```

워크플로우 클래스의 새 인스턴스가 만들어지면, Windows Forms 을 사용해서 정의한 양식 구성자가 컨트롤을 초기화시키듯이, 구성자는 그 워크플로우 안에 액티비티를 만들고 구성한다.

XML-based eXtensible Application Markup Language (XAML)을 사용해서도 워크플로우를 정의할 수 있다. XAML 의 워크플로우 클래스 기본 코드 뼈대는 다음과 같을 것이다:

```
<?Mapping XmlNamespace="Activities"
   ClrNamespace="System.Workflow.Activities"
   Assembly="System.Workflow.Activities" ?>
<SequentialWorkflow x:Class="ExampleWorkflow"
   xmlns="Activities" xmlns:x="Definition">
    …
</SequentialWorkflow>
```

이 예제를 시작하는 **Mapping** 처리 명령은 코드 예제를 시작하는 **using** 문과 같은 것이다. 두 명령 모두 사용할 네임스페이스를 지정하기 때문이다. 그 후에 **SequentialWorkflow** 요소는 이 네임스페이스를 사용하는 **ExampleWorkflow** 라는 클래스를 정의한다.

워크플로우 정의를 위해 왜 마크업 옵션도 제공하는 것일까? 한 가지 이유는 특정 애플리케이션 개발을 위해 이런 옵션을 선호하는 개발자도 있다는 것이다. 또 다른 이유는 많은 툴 개발업체들이 코드를 만들고 파싱하는 것보다 XML 을 만들고 파싱하는 툴을 더 쉽게 만들 수 있다는 것이다. 실제로, WF 의 Workflow Designer 는 XAML 을 만들기 때문에, 이 툴을 사용하는 개발자들은 XML 버전의 워크플로우를 보게 된다. Workflow Designer 산출물, 개발자의 코드와 XAML 를 결합해서 워크플로우를 만들 수 있다. 워크플로우를 어떻게 만들던, 결국에는 모든 워크플로우가 표준 .NET 어셈블리로 컴파일된다.

액티비티 만들기

액티비티는 워크플로우의 빌딩 블록이다. 한 동작은 기본 액티비티 라이브러리의 Delay 와 Terminate 같이 작고 기술에 집중된 것이거나 WF 기반으로 ISV 가 정의하는 ProcessHelpDeskRequest 액티비티와 같이 더 크고 비즈니스에 집중된 것일 수 있다. 액티비티가 간단하건 복잡하건, 마이크로소프트가 만들었던 다른 벤더가 만들었던 상관없이, 모든 액티비티는 같은 WF 인터페이스를 사용해서 구현된다.

\새 액티비티는 처음부터 만들거나 기존 액티비티를 확장할 수 있다. 새 액티비티는 기존 액티비티를 복합 액티비티 (composite activity)으로 묶어서 만들 수도 있다. IfElse, Sequence, While, TransactionScope 와 같이 기본 액티비티 라이브러리에 있는 많은 액티비티들은 실제로는 복합 액티비티다. 실제로, 워크플로우 자체는 특수한 종류의 복합 액티비티일 뿐이다.

액티비티를 만드는 가장 간단한 방법은 그래픽 Activity Designer 툴을 사용하는 것이다. 액티비티는 워크플로우와 마찬가지로 클래스이기 때문에 코드로 직접 만들 수도 있다. 각각의 액티비티는 대응하는 이벤트와 액티비티 상태를 가지는 속성을 가질 수 있으며 C#으로 정의한 액티비티는 다음과 같다:

```
using System.Workflow.ComponentModel;
public class ExampleActivity : Activity
{
  …
}
```

액티비티 개발자가 이것을 어떻게 만들던 상관없이, 그 액티비티를 사용하는 다른 개발자에게 액티비티의 내부를 얼마나 많이 보여줄 것인지를 결정할 수 있다. 이렇게 되면 전용 정보를 통제할 수 있게 된다. 액티비티 개발자는 사용자 지정 디자이너 테마를 정의해서 Workflow Designer 에서 액티비티를 그래픽으로 표현하는 것과 같이, 액티비티의 룩앤필을 결정할 수 있다. 디자이너 테마를 정의하면 ISV 들이 자신의 애플리케이션에 WF 를 쉽게 구현할 수 있는 동시에 자신의 제품의 스타일을 그대로 유지할 수 있게 된다.

조건과 규칙 사용하기

비즈니스 프로세스는 비즈니스 규칙을 따른다. 예를 들어 한 기업이 구매 관리자가 1 억 원까지의 구매 요청을 승인하거나, 고객에게 10% 할인을 해주거나, 직원이 하루에 300 달러가 넘지 않는 방에서 숙박할 수 있게 할 수 있다. 이런 간단한 문장은 비즈니스 규칙을 표현하고 있는 것이다. 비즈니스 규칙을 제대로 처리하는 것이 워크플로우 개발에 있어서 중요한 부분이며, WF 는 이 같은 어려움을 해결하는 다양한 방법을 제공하고 있다.

간단한 조건 정의하기

전형적인 워크플로우에서는, IfElse 또는 While 액티비티에 있는 각각의 조건은 실제로는 비즈니스 규칙이다. WF 는 이런 조건을 정의할 수 있는 두 가지 옵션을 제공하고 있다. 한 옵션은 코드에서 조건을 직접 구현하는 것으로 코드 조건 (code condition)으로 부른다. 개발자는 관련 데이터를 평가하고 Boolean 값을 반환하는 메서드를 작성해서 이 조건을 구현한다. 조건을 평가해야 할 경우, 이 메서드가 호출되고 반환되는 결과값이 사용된다.

다른 옵션은 코드 또는 Rule Condition Editor 를 사용해서 규칙 조건 (rule condition)을 정의하는 것이다. 규칙 조건은 별도의 파일에 XML 로 저장된다. 워크플로우가 규칙 조건에 도달하면, 그 조건의 수식이 평가되고 Boolean 값이 반환된다. 규칙 조건은 코드 조건과 달리, 실행 중인 워크플로우를 바로 변경할 수 있다. 같은 규칙 조건이 같은 워크플로우의 여러 액티비티에 사용될 수 있기 때문에 비즈니스 규칙이 한 곳에서만 표현되면 된다.

조건과 액티비티를 하나의 그룹으로 만들기: CAG Activity

IfElse 와 While 과 같은 액티비티는 워크플로우가 실행하는 액티비티들을 쉽게 통제할 수 있는 방법을 제공한다. 사람이 개입하는 워크플로우와 같이 더 동적인 시나리오에 대해서는, WF 의 Conditioned Activity Group (CAG) 액티비티가 많은 도움이 된다. CAG 액티비티는 다른 액티비티도 가지고 있는데, 각각의 액티비티는 when 조건이라고 알려진 것과 관련이 있다. 이 액티비티는 코드 조건 또는 규칙 조건을 사용해서 when 조건을 지정할 수 있다.

워크플로우에서 CAG 액티비티를 만나게 되면, 모든 when 조건이 평가된다. 참을 반환하는 모든 when 조건은 관련 액티비티가 실행되게 만든다. 이런 액티비티는 복합 액티비티일 수 있기 때문에, 이 실행은 상당히 복잡할 수 있다.

CAG 의 액티비티는 병렬로 실행되고 이 모든 액티비티가 완료되면, 현재 실행되지 않고 있는 CAG 의 모든 액티비티에 대해 when 조건이 다시 한 번 평가된다. 다시 한 번 더 참으로 평가되는 모든 when 조건은 자신과 관련된 액티비티가 실행되게 한다. when 조건을 확인하고 참으로 평가된 조건과 관련된 액티비티를 실행하는 사이클은, 더 이상 when 조건이 참이 아닌 경우 또는 CAG 액티비티와 관련된 until 조건이 참이 되는 경우까지 계속 된다.

CAG 액티비티는 비즈니스 규칙에 따라 진행되는 미니 워크플로우로 생각할 수도 있다. which 조건이 참인 경우에는 액티비티 그룹을 계속 실행하기 때문이다. 이것은 규칙 엔진을 사용하는 것과 비슷하다. WF 는 완전한 기능을 제공하는 규칙 엔진이 필요한 애플리케이션도 지원한다.

규칙 엔진 사용하기: Policy Activity

복잡한 비즈니스 규칙을 평가하기 위해, 복잡한 조건을 만들고 유지하는 것은 상당히 어려운 일이다. 앞에서 설명한 보험 애플리케이션 프로세스에서는, 회사의 정책을 구성하는 매우 복잡한 규칙들에 대해 모든 신청자를 확인해야 한다. 앞에서 설명했던 WF 옵션을 사용해서 이런 규칙들을 표현할 수 있지만, 항상 최선의 방법이 될 수는 없다. 대신에, 규칙 엔진을 사용하는 것이 복잡한 규칙 그룹을 처리하는 올바른 방법이 될 수도 있다.

WF 는 Policy 액티비티를 통해 접속할 수 있는 규칙 엔진을 제공하고 있다. 개발자는 이 액티비티를 사용해서, 규칙 집합 (rule set)이라는 규칙 그룹을 정의할 수 있다. 각각의 규칙은 IF <condition> THEN <action> ELSE <action>의 형태를 가진다. 예를 들면, 보험 회사는 승인되는 모든 자격을 가지고 있는 규칙 집합으로 Policy 액티비티를 만들 수 있다. 예를 들면 21 세 미만의 운전자가 대학생이 아니라면 높은 위험으로 분류하거나 기혼 운전자는 낮은 위험을 분류할 수도 있다. 특정 신청자가 이런 규칙을 준수하는 지를 판단해야 하는 워크플로우는 Policy 액티비티를 실행시킬 것이다. WF 규칙 엔진은 이 규칙 집합에서 어느 규칙이 참인지를 판단하고, 그 규칙을 실행한다. 이 같은 실행으로 규칙 집합의 다른 규칙들이 참이 되도록 워크플로우의 상태가 변경된다. 규칙 엔진은 규칙 집합에서 영향을 받는 모든 규칙들을 검사하고 다시 실행한다. 이 기법은 전방향 추론 (forward chaining)이라고 부른다. 이 프로세스는 어떤 새 규칙도 참이 되지 않거나 사전에 정의된 한계에 도달할 때까지 계속 된다.

Policy 액티비티를 사용하려면 약간의 코드를 작성해야 한다. 예를 들어 Workflow Designer 툴박스에 이 액티비티에 대한 그래픽 표현이 없다면, 개발자는 특정 Policy 액티비티를 명시적으로 만들어야 한다. 또한 개발자는 규칙 집합에 그 규칙을 정의하는 코드도 작성해야 한다.

런타임 엔진 호스팅하기

모든 워크플로우는 WF 런타임 엔진을 사용한다. 이 엔진은 각각의 워크플로우를 실행하고 워크플로우가 완료될 때까지 상태를 관리한다. WF 런타임 엔진은 라이브러리이기 때문에 호스트 프로세스에서 실행되어야 한다. WF 는 한 호스트를 지정하지 않고, 런타임 엔진 (그리고 런타임 엔진이 실행하는 모든 워크플로우)이 간단한 콘솔과 Windows Forms 애플리케이션부터 복잡한 워크플로우 전용 서버에 이르기 까지 거의 모든 윈도우 프로세스에서 호스팅될 수 있게 한다. WF 는 런타임이 ASP.NET 애플리케이션에서 실행될 수 있게 하는 서비스들을 함께 제공하고 있지만, ISV 와 일반 사용자들은 자신의 호스트를 자유롭게 만들거나 기존 애플리케이션에서 WF 런타임 엔진을 호스팅할 수 있다. Windows SharePoint Services 와 System Center Service Manager 등의 다른 마이크로소프트 제품들도 WF 런타임 엔진을 호스팅할 수 있다.

각각의 호스트는 나름대로의 특징을 가지고 있다. 예를 들면, 간단한 Windows Presentation Foundation 애플리케이션에서 실행되는 워크플로우는 Windows SharePoint Services 에서 호스팅된 워크플로우보다 확장성과 안정성이 낮을 것이다. WF 는 스탠드얼론 제품이 아니라 워크플로우 프레임워크이기 때문에, 이렇게 다양한 특징을 지원하도록 설계되었다. 모든 호스트가 같은 런타임 엔진을 사용한다고 해도, 각각의 호스트는 앞의 다이어그램과 같은 런타임 서비스들을 제공해야 한다. 이 서비스들은 워크플로우의 상태를 저장하고, 워크플로우의 실행을 추적하고, 트랜잭션을 사용하는 기능을 지원한다. WF 는 모든 호스트에서 사용할 수 있는 이 모든 서비스를 기본으로 제공한다. 그렇지만 호스트가 독특한 요구사항을 충족시키도록 사용자 지정하려면, 호스트를 만든 사람은 필요에 따라 기본값을 바꿀 수 있다.

WF 런타임이 호스트가 제공하는 서비스를 어떻게 처리하는 지를 알려면, 워크플로우의 기본 특징 중 하나인, 오랜 시간 실행될 수 있다는 특징에 대해 생각하는 것이 좋다. 워크플로우는 몇 시간, 며칠 또는 몇 주 동안 실행될 수 있기 때문에, 워크플로우가 오랜 시간 동안 동작하지 않는다면 WF 런타임은 자동으로 실행 중인 워크플로우를 종료시키고 그 상태를 저장할 것이다. 이 프로세스를 dehydration 이라고 부르기도 한다. 워크플로우가 외부 이벤트를 기다리면서 멈춰져 있기 때문에 런타임 엔진이 일반적으로 워크플로우를 내리는 결정을 한다. 그렇지만 그 워크플로우의 상태를 디스크에 저장하기 위해, 런타임 엔진은 호스트 서비스가 제공하는 영속성 (persistence) 서비스를 사용해야 한다. WF 와 함께 제공되는 ASP.NET 기반의 호스트는 SQL Server 또는 무료로 배포가능한 SQL Express 을 활용한다. 또 다른 호스트는 ISV 애플리케이션에서 사용하는 전용 데이터베이스와 같은 다른 기술을 사용해서 워크플로우를 저장할 것이다.

호스트가 어떤 서비스를 제공하던, WF 런타임을 호스팅하는 것은 다음과 같이 간단하다:

```
using System.Workflow.Runtime;
```

```
class ExampleHost
{
 static void Main()
 {
   WorkflowRuntime runtime = new WorkflowRuntime();
   runtime.StartRuntime();
   runtime.StartWorkflow(typeof(ExampleWorkflow));
   …
 }
}
```

이 예에서와 같이, 런타임 엔진 역시 또 하나의 클래스다. 이 클래스의 인스턴스가 만들어지면, StartRuntime 메서드를 호출해서 초기화시킨 후에, StartWorkflow 메서드를 통해서 워크플로우가 이 인스턴스를 실행시키게 한다.

호스트 프로세스는 워크플로우가 실행되는 환경에서 중요한 부분이다. 런타임은 다양한 윈도우 프로세스에서 호스팅될 수 있기 때문에, WF 워크플로우는 매우 다양한 시나리오에서 사용될 수 있다.

워크플로우 외부의 소프트웨어와 통신하기

일반적인 워크플로우의 역할이 사람과 시스템의 작업을 조율하는 것이기 때문에, WF 는 워크플로우가 다른 소프트웨어와 통신할 수 있는 방법을 제공해야 한다. 대기 중인 워크플로우는 내려지고, 그 상태는 디스크에 저장된다는 것을 기억하고 있을 것이다. 이런 상황의 워크플로우는 실행되고 있지 않기 때문에 수신되는 모든 통신을 직접 수신할 수 없다. 이런 가능성에 대해, WF 런타임은 모든 워크플로우의 모든 통신에 대해 전도체 역할을 한다. 요청이 수신되면, 런타임 엔진이 이것을 수신하고 이 요청이 어느 워크플로우 인스턴스에게 향하는 것인지를 판단한다. 그 후에 런타임 엔진은 대상 인스턴스에 요청을 전달하는데, 그 인스턴스가 저장되어 있는 상태라면 그 인스턴스를 먼저 다시 로딩한다. 실제로, WF 런타임은 워크플로우 외부의 소프트웨어와의 모든 통신에 대해 프록시 역할을 한다.

워크플로우는 같은 윈도우 프로세스의 다른 객체와 통신하기 위해, 베이스 액티비티 라이브러리의 두 액티비티를 사용한다. CallExternalMethod 액티비티는 워크플로우 외부의 객체의 메서드를 호출할 수 있게 하는 반면에, HandleExternalEvent 액티비티는 워크플로우 외부의 객체의 호출을 받을 수 있게 한다. 개발자는 인터페이스를 정의해서 각각의 호출에 대해 이름과 패러미터를 지정할 수 있다.

WF 는 웹 서비스를 통해 통신할 수 있는 액티비티도 내장하고 있다. ASMX 로 알려진 ASP.NET Web services 에 구현된 액티비티들은 다음과 같다:

InvokeWebService: 웹 서비스를 호출하고 응답을 동기시켜 반환한다.

WebServiceReceive: 수신되는 웹 서비스 호출을 수용한다.

WebServiceResponse: WebServiceReceive 를 통해 수신된 웹 서비스에 대한 응답을 전송한다.

워크플로우는 WebServiceReceive 와 WebServiceReponse 를 사용해서, 웹 서비스를 통해 클라이언트에게 자신을 공개한다. 이렇게 되면, 워크플로우는 웹 서비스를 제공하는 동시에 소비할 수 있게 된다. .NET Framework 3.5 와 함께 제공되는 WF 부터는, 워크플로우가 WCF 를 사용해서 외부와 통신할 수 있다. 다음은 외부와 통신하는 방법에 대해 설명하고 있다.

워크플로우가 사용하는 서비스 만들기

WF 와 WCF 는 결합되어 사용되고 있다. 워크플로우는 서비스를 실행시켜야 하며, 워크플로우와 함께 서비스를 구현하는 것이 좋은 생각이다. .NET Framework 3.0 의 WF 에서는 이 두 기술의 결합이 쉽지 않았지만, .NET Framework 3.5 에서는 크게 달라졌다. 이 버전에서는, WF 와 WCF 를 사용해서 워크플로우가 사용하는 서비스를 쉽게 만들 수 있다.

두 개의 새로운 WF 액티비티를 통해 결합된다:

Send: WCF 를 사용해서 요청을 전송한 후에, 선택적으로 응답을 기다린다. 개발자는 실행되어야 할 연산작업과 그 연산작업을 발견할 수 있는 엔드포인트를 지정한다. 웹 서비스 또는 WCF 가 제공하는 다른 옵션을 사용해서 통신을 한다.

Receive: WCF 를 통해 수신되는 요청을 받고, 응답을 전송한다. 개발자는 이 요청을 수용하는 연산작업만을 지정한다. Receive 는 복합 액티비티로 자신이 가지고 있는 액티비티를 통해 연산작업의 모든 로직을 구현한다. 이 액티비티를 사용해서 실행되고 있는 워크플로우가 수신되는 요청을 기다리게 하거나, 요청이 수신되면 Receive 액티비티로 시작되는 워크플로우가 새 인스턴스를 만들게 할 수 있다. 그리고 Send 에서와 같이, 웹 서비스 또는 WCF 가 제공하는 또 다른 옵션을 사용해서 통신을 한다.

이 두 액티비티를 WF 워크플로우로 드랙앤드롭한 후에 필요에 따라 구성할 수 있다. 워크플로우 작성자는 WCF 통신의 모든 것 – 바인딩, 컨트랙트, 보안 옵션 등 – 을 설정할 수 있다. Visual Studio 2008 은 워크플로우가 사용하는 서비스를 만들 수 있는 프로젝트 형을 제공해서 개발자의 부담을 덜어준다.

워크플로우 실행 추적하기

워크플로우는, 잘 정의된 실행 단위인 액티비티에서 만들어진다. 이 공용 구조는 표준 방식으로 모든 워크플로우의 실행을 추적할 수 있다. WF 는 워크플로우가 실행을 시작하고 종료할 때, 워크플로우 내의 액티비티가 들어왔다가 나가는 때와 같은 워크플로우의 추적 정보를 제공할 수 있다. 이 기능은, 주문처리 시간 측정 또는 실행 중인 모든 워크플로우에 대한 정보를 가진 데이터베이스 관리, 이 데이터를 실시간으로 질의하기 등의 워크플로우의 기술 또는 비즈니스 부분을 수집하는데 많은 도움이 된다. 개발자가 정의한 프로파일에 따라 정확하게 어떤 추적 정보가 만들어지는 지가 결정되고 그 정보는 기본적으로 SQL Server 데이터베이스에 기록된다. 그렇지만 추적 기능은 워크플로우 애플리케이션에서 교체가 가능한 런타임 서비스로 구현되기 때문에, 이런 기본 설정을 바꿀 수 있다.

WF 의 추적 기능은 비교적 간단하다. 추적 기능은 추적 데이터를 정의하고 수집하는데 필요한 인터페이스를 제공하지만 이 정보를 보여주는 툴은 제공하지 않는다. 추적 데이터에 따라 통지를 하는 표준 메커니즘은 없다. WF 는 BizTalk Server 의 Business Activity Monitoring 와 같은 완벽한 서비스를 제공하는 대신에, ISV 와 개발자가 이용할 수 있는 인프라를 제공하고 있다.

실행 중인 워크플로우 변경하기

WF 가 소프트웨어 간의 연동을 조율하는 시스템 워크플로우에만 집중한다면, 워크플로우를 정의하고 이것을 변경하지 않고 실행할 수 있는 능력만으로도 충분할 것이다. 그렇지만 사람들은 자신의 비즈니스 프로세스가 더 동적이기 바라기 때문에 휴먼 워크플로우를 지원하려면 실행 중에 변경할 수 있는 능력이 필요하다. 현재 실행되고 있는 워크플로우 참여자 중 한 사람이 비즈니스 프로세스에 새 단계를 추가하기로 결정했다고 가정해보자. 아마도 앞에서 설명했던 보험 애플리케이션 프로세스의 관리자는 높은 위험의 신청자에 대해 다른 의견을 얻고 싶어할 수도 있는데, 이것은 실행 중인 윈도우 인스턴스에 또 하나의 단계를 추가하는 것이다. 워크플로우 개발자가 좋아하던 말던, 자동화된 비즈니스 프로세스를 사용하는 사람들은 이런 변경을 할 수 있기를 바란다.

이런 요구사항에 대해, WF 는 다이나믹 업데이트 기능을 포함하고 있다. 이 옵션을 사용해서, 실행 중인 워크플로우의 인스턴스는 작성자가 지정한 안전하고, 잘 정의된 경계 내에서 수정된 후에 새 워크플로우로 저장될 수 있다. 예를 들어, 새 활동은 워크플로우 또는 IfElse 나 CAG 활동에 대해 변경된 규칙 조건에 삽입될 수 있다. 시스템 지향적인 시스템에서는 이런 실행 중 업데이트가 발생되지 않지만, 사람의 활동을 조율하는 워크플로우에서는 이것이 반드시 필요하다.

휴먼 워크플로우 지원하기

이미 설명한 것과 같이, 휴먼과 시스템 워크플로우를 모두 지원하는 것이 WF 의 주요 목적이다. 개발자가 사람과 관련된 워크플로우를 만들기 위해 사용할 대부분의 기술은 다음과 같이 이미 모두 설명되었다:

상태 머신 워크플로우. 순차 워크플로우를 사용하는 것보다 더 동적이고 이벤트 중심 방식으로 워크플로우 단계들을 정의.

CAG 활동. 활동 그룹이 규칙에 따라 실행.

다이나믹 업데이트. 사용자가 실행 중인 워크플로우를 변경.

또 다른 기본 활동 라이브러리의 멤버인 Replicator 활동도 설명할 필요가 있다. 이 이름이 알려주듯이, 이 활동은 지정된 횟수만큼 모든 활동을 복제할 수 있다. 예를 들면, 워크플로우에서 Replicator 를 사용해서, 관리자가 자신의 팀원에게 특정 과업을 할당할 수 있게 한다. Replicator 활동은 각각에 대해 그 과업의 활동의 인스턴스를 만들어준다. 런타임에서 복제 회수가 지정될 수 있게 해서, Replicator 활동은 워크플로우가 비즈니스 프로세스에 관련된 사람들에게 작업을 할당할 수 있는 동적인 방법을 제공한다.

사람들이 워크플로우를 사용하는 방법과 관련된 WF 의 부분은 역할 (roles)에 대한 지원이다. WF 는 사용자의 역할에 따라 특정 활동을 실행할 수 있는 권한을 부여하는 범용 인프라를 구현하고 있다. 워크플로우 개발자는 그 워크플로우에서 사용되는 역할을 정의하고, 어느 역할이 어느 활동을 수행할 수 있는 지를 결정한다. 예를 들면 Applicant 역할인 사람은 보험 신청을 할 수 있지만, 보험 애플리케이션은 Manager 역할인 사람만이 승인할 수 있다. 그리고 WF 는 역할을 관리하기 위해 자체 시스템을 구현하는 대신에 워크플로우가 외부의 역할 관리 시스템을 사용할 수 있게 하는 메커니즘을 제공한다. WF 는 윈도우 계정, Active Directory, ASP.NET 을 사용해서 정의한 역할을 지원하는 기능을 내장하고 있지만 다른 역할 관리 시스템을 사용할 수도 있다.

WF 는 다양한 종류의 애플리케이션에서 사용할 수 있는 범용 워크플로우 프레임워크를 제공한다. 이 프레임워크가 현재의 윈도우 플랫폼에 있어서 표준이기는 하지만, WF 가 마이크로소프트의 다른 기술과 어떻게 어울릴 수 있는지 궁금할 것이다. 여기에서는 WF 가 BizTalk Server, Windows SharePoint Services, 2007 Microsoft Office system 과 같은 마이크로소프트의 다른 제품들과 어떻게 연동되는지를 설명하고 있다.

Windows Workflow Foundation 과 BizTalk Server

아마도 가장 유명한 마이크로소프트의 워크플로우 기능은 BizTalk Server 에 있을 것이다. 개발자는 BizTalk Server 를 통해 business process management (BPM), enterprise application integration (EAI), business-to-business (B2B) 통합의 오케스트레이션 (시스템 워크플로우)을 만들 수 있다. BizTalk Server 2006 R2 다음 버전에서는 이런 영역에 대해 WF 워크플로우를 만들 수 있는 지원 기능이 추가될 예정이다.

BizTalk Server 와 WF 는 일부 비슷한 면을 가지고 있다. 개발자에게는, BizTalk Server Orchestration Designer 는 WF 의 Workflow Designer 와 상당히 비슷해 보일 것이다. 마이크로소프트의 한 개발 그룹이 두 기술을 개발하고 있기 때문에 당연한 것이다. 그렇지만 특정 문제에 어떤 기술을 사용할 것인지를 쉽게 결정할 수 있다. 어느 기술을 사용할 것인지는 다음 가이드라인에 따라 결정하면 된다.

WF 를 사용하는 경우:

애플리케이션이 워크플로우를 호스팅한다. WF 는 자신의 프로세스에서 실행되는 워크플로우 기반의 애플리케이션을 만들 수 있다. BizTalk Server 는 범용 워크플로우 프레임워크를 제공하지 않고 다양한 애플리케이션을 통합하는 것에 초점을 맞추고 있기 때문에, BizTalk Server 프로세스 내에서 오케스트레이션을 실행한다.

휴먼 워크플로우가 필요한 프로세스를 구현한다. BizTalk Server 는 시스템 워크플로우를 지원하기 때문에, WF 의 상태 시스템 워크플로우와 다이나믹 업데이트와 같은 지원을 하지 못한다. 휴먼 워크플로우와 더 복잡한 시스템 통합 서비스 모두가 필요한 시나리오는 WF 와 BizTalk Server 모두를 사용해서 구현할 수 있다. 예를 들면, Windows SharePoint Services 가 지원하는 휴먼 워크플로우는 문제의 인간 영역에 사용되는 반면에, BizTalk Server 는 시스템 통합 영역을 처리한다. 두 영역은 BizTalk Server Adapter for SharePoint 를 사용해서 연동된다.

워크플로우는 클라이언트 시스템에서 실행된다. BizTalk Server 는 서버 중심의 제품이기 때문에 데스크톱 컴퓨터에서는 어울리지 않는다.

BizTalk Server 를 사용하는 경우:

전형적인 EAI 시나리오에서와 같이, 다양한 플랫폼에서 다양한 애플리케이션과 통신해야 하는 문제를 해결한다. BizTalk Server 는 플랫폼 간의 통합에 초점을 맞추고 있어서 다양한 소프트웨어와의 통신을 위한 많은 어댑터가 제공된다. WF 는 EAI 가 아닌 워크플로우에만 초점을 맞추고 있어서 이런 기능은 제공하지 않는다.

B2B 서비스가 필요하다. WF 는 이 영역을 지원하지 않지만, BizTalk Server 는 거래 파트너와의 작업용 툴인 Electronic Data Interchange (EDI), RosettaNet, SWIFT 와 다른 산업표준용 액셀러레이터를 지원한다.

Business Activity Monitoring (BAM)과 같은 BPM 서비스가 필요하다. WF 추적 인프라를 사용해서 이런 서비스를 만들 수 있지만, BizTalk Server 는 정보 근로자가 BAM 뷰를 정의할 수 있는 툴과 같이, 더 완벽한 솔루션을 제공한다.

완벽한 관리 인프라와 높은 확장성 지원이 필요하다. WF 와 달리, BizTalk Server 는 제품 환경을 관리하고 확장시킬 수 있는 완벽한 툴들을 제공하고 있다.

Windows Workflow Foundation 과 Windows SharePoint Services

Windows Server 운영체계의 표준 기능인 Windows SharePoint Services 는 사람과 소프트웨어 더 효과적으로 협업할 수 있게 한다. Windows SharePoint Services 최신 버전은 당연히 WF 를 호스팅한다.

개발자는 Workflow Designer 와 같은 표준 WF 툴을 사용해서, 문서 협업과 다른 종류의 정보 공유를 지원하는 워크플로우 애플리케이션을 만들 수 있다. 기술력이 높지 않은 사람들은 Office SharePoint Designer 툴을 사용해서 워크플로우를 만들 수 있다. 이 툴은 Workflow Designer 의 그래픽 기술을 사용하지 않고 규칙 기반의 방식을 통해 워크플로우를 지정한다. Windows SharePoint Services 의 WF 지원은 2007 Office system 에서 워크플로우를 사용할 수 있는 인프라도 제공한다.

2007 Office system 에는 Word, Excel, Outlook, PowerPoint 와 같은 오피스 데스크톱 애플리케이션이 포함되어 있다. 또한 다양한 분야를 지원하는 별도의 라이선스를 가지고 있는 서버들이 포함되어 있다. 워크플로우와 가장 관련있는 서버는 Office SharePoint Server 2007 로, Windows SharePoint Services 에 내장된 다음과 같은 WF 기반의 워크플로우 추가 기능을 제공하고 있다:

미리 정의된 워크플로우 그룹. 이 워크플로우들은 정보 근로자가 직접 사용할 수 있는 것으로, 다양한 요구사항에 맞도록 사용자 지정될 수 있다. Office SharePoint Server 가 제공하는 미리 정의된 워크플로우는 문서 승인을 위해 라우팅하는 워크플로우, 참여자가 기한 만료된 문서를 보유하거나 삭제할 것인지를 결정할 수 있게 해서 문서 보유를 관리하는 워크플로우 등이 있다.

Office 2007 클라이언트 지원. Windows SharePoint Services 를 사용해서 만든 워크플로우는 ASPX 양식을 사용해야 하기 때문에, 사람들이 이 워크플로우를 사용할 수 있는 방법은 웹 브라우저뿐이다. Office SharePoint Server 에서 실행되는 Windows SharePoint Services 워크플로우는 브라우저 대신에 Office 2007 애플리케이션에서 사용자가 자신의 양식을 사용할 수 있게 지원한다.

InfoPath 를 사용해서 양식을 정의할 수 있는 워크플로우 능력. 대부분의 사람들은 ASPX 페이지보다 InfoPath 양식을 사용하는 것이 더 쉬우며, 이 양식은 검증과 같은 기능을 내장하고 있다.

Office 데스크톱 제품의 인기를 감안하면, 오피스 사용자가 친숙한 오피스 인터페이스를 사용해서 워크플로우를 사용하는 것이 당연하다. WF 를 2007 Office system 과 통합해서 사용자가 이 인터페이스를 사용할 수 있다.

결론

올바른 인프라없이 워크플로우를 개발하는 것이 쉽지 않다. 오랜 시간 실행되는 비즈니스 프로세스의 요구를 충족시키고, 사람들이 요구하는 다이나믹한 동작을 지원하고, 워크플로우가 가진 다른 문제들을 해결하려면 일반 애플리케이션보다 더 많은 시간과 노력이 필요하다. 올바른 지원 기술이 제공된다면, 워크플로우를 쉽게 만들 수 있으며 워크플로우 소프트웨어가 훨씬 더 많이 사용될 것이다.

WF 의 목적은 이런 인프라를 제공하는 것이다. 휴먼과 시스템 워크플로우를 모두 지원하는 범용 프레임워크를 제공하고, 이 프레임워크를 윈도우의 표준 기능으로 통합해서 워크플로우 기술이 광범위하게 사용될 수 있는 토대를 마련하였다. 특수 상황에만 사용되던 워크플로우는 이제 표준 기술이 되어 가고 있다.

저자 소개

David Chappell 은 Chappell & Associates (www.davidchappell.com)의 대표로, 강연, 저술과 컨설팅을 통해 전세계 IT 전문가들이 기업용 소프트웨어를 보다 잘 이해하고 사용하도록 돕고 있다.