2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005 년 5 월 13 일~14 일, 충북대학교

프로세스간 상호운영을 가능하게 하는 효과적인 SCM 실행방법론
An Efficient Operation Method Enabling SCM Process Interoperability

이승현*, 배혜림**

* 서울대학교 산업공학과
** 부산대학교 산업공학과

초 록

기업은 날이 갈수록 심화되는 경쟁 환경에 대비하기 위해 많은 공급업체들과의 협력관계를 점차 강화해나가고 있다. 정보기술이 발전함에 따라 기업들은 SCM (Supply Chain Management)을 도입하여 이를 실현해나가고 있는 추세이다. 이에 따라, 공급사슬상의 관련 기업간의 상호작용과 정보공유에 기반한 공급사슬 통합이 추진되고 있으며, 정교한 관리를 위해 프로세스 간의 통합을 통해 하나의 글로벌 프로세스로서 SCM을 관리할 필요가 대두되고 있다. 이러한 관점에서 프로세스를 하나의 객체로서 관리하는 BPM (Business Process Management)을 활용하여 SCM을 체계적으로 관리할 필요가 있다. 그러나, 지금까지의 BPM은 비즈니스 프로세스의 자동화 및 통제 등에 주력해왔기 때문에, 전체 프로세스 관점에서 이를 효율적으로 실행하는 기능을 제공하지 못하고 있다. 따라서, BPM을 통해 SCM 프로세스를 단순히 자동화하여 실행한다면, 각 참여 업체들이 글로벌 프로세스의 관점에서 효율적으로 업무를 수행하기 어렵다. 본 논문에서는, 각 공급업체들이 전체 프로세스 관점에서 효율적으로 업무를 수행하는 방법을 제안한다.
제안한 방법을 통해 각 업체들은 전체 프로세스 내 관련 경로의 긴급도 및 관련 업무의 우선처리순위 정보를 제공 받을 수 있다. 따라서, 관련 정보를 토대로 각자의 업무를 수행하기만 하면, 글로벌 SCM 프로세스의 효율성을 향상시킬 수 있을 것이다.

주제어: BPM, BPEL, SCM, Dispatching Rule

1. 연구의 개요

기업들은 경쟁력을 강화해가기 위해, 다른 업체들과의 협력을 필요로 하고 있다. 이러한 협력 체계는 이미 오래 전부터 이루어져 왔지만, 기술적, 문화적인 이유 등으로 시스템에 의해 구현되어 오지 못했다. 최근 정보시스템 시장에서 가장 주목 받고 있는 개념인 BPM (Business Process Management)과 이를 구현할 수 있는 언어로서 BPEL (Business Process Execution Language)을 활용하여 기업의 협력 체계를 구현할 수 있게 되었다[배2004]. 그렇지만, 이기종 환경에서 운영되는 프로세스 통합을 구현하는 것만으로도 많은 시간과 노력이 투입되기 때문에, 상당 수의 경우 프로세스 통합 이후 이를 효율적으로 관리하는 방법까지 제공하지 못하고 있다. 따라서, 본 논문에서는 BPM 실행프로세스의 효율적인 관리방법에 관한 기존 연구[Rhee2004]를 응용하여 중첩프로세스를 포함하고 있는 SCM (Supply Chain Management) 프로세스의 효율적인 관리 방안을 제안한다.

2. 연구의 배경

2.1 BPM 개요

BPM (Business Process Management)은 기업의 생산성을 제고하기 위해 업무절차를 체계적으로 설계, 관리, 개선하는 활동을 지원하는 관리 방법론 또는 이를 지원하는 소프트웨어 시스템으로 정의한다[Smith2003]. (단, 본 논문에서 언급하는 BPM은 소프트웨어 시스템을 뜻한다.) BPM은 프로세스의 설계, 실행, 모니터링, 분석의 기능을 수행하고, 외부 시스템 호출 또는 연동 기능을 지원한다[Rhee2005]. 특히, 이기종 시스템 간 상호운영성을 지원함으로써, 서로 다른 기업의 상호작용과 통합의 도구로서 사용되고 이는 BPM을 구축함으로써 얻을 수 있는 장점 중 하나로 알려지고 있다[Rhee2005]. 즉, 서로 다른 환경에서 실행되는 개별 비즈니스 프로세스를 하나의 글로벌 프로세스로 구성하고 이를 실행하는데 있어 발생하는 기술적 문제를 해결한다[Smith2003]. 최근에는 여러 기업이 상호작용하면서 프로세스를 진행하는 SCM, B2Bi 등에 BPM이 활발하게 도입되면서 프로세스 실행 자체의 효율성 제고는 물론이고, 체계적인 관리, 감독을 통한 지속적 개선이 이루어지고 있다.
비즈니스 프로세스를 통합 관리하는 BPM의 핵심

기능은 워크플로우(Workflow)기술을 통하여 구현되며, 워크플로우 프로세스는 일반적으로 정의시(build-time) 단계와 실행시(run-time) 단계로 나누어 생각하는 것이 일반적이다[Rhee2004]. 정의시 단계는 프로세스 수행을 준비하는 과정으로, 프로세스를 이루는 단위업무와 이들 간의 선후 관계 그리고 각 단위업무를 수행하는데 필요한 세부 속성들을 정의하며 이에 대해서는 2.2절에서 자세히 설명하기로 한다. 실행시 단계에서는 정의시 단계의 프로세스 모델을 해석하여 실제 프로세스를 실행, 관리, 통제한다[Kim2000]. BPM에서 실행시 단계를 담당하는 부분을 워크플로우 엔진 (workflow engine)이라 한다[Workflow1999]. 실제로 처리해야 할 프로세스가 발생하면 엔진은 해당되는 정의시의 프로세스 모델을 바탕으로 실제 프로세스를 발생시키게 되는데, 이 때 생성되는 개별 프로세스를 프로세스 인스턴스(instance)라고 한다[Kim2000].

2.2 프로세스 모델

프로세스 자동 수행을 위해서는 컴퓨터 시스템이 이해할 수 있는 형태로 프로세스가 표현되어 있어야 한다. 본 논문에서는 이것을 ‘프로세스 모델(*Process Model*)’이라고 한다. 여기에는 프로세스가 추구하는 목적 달성에 필요한 업무와 이들간의 상호관계가 명세 되어 있다[Bae2004]. 본 논문에서는 연구에서 고려하는 소기의 목적을 달성하는데 필요한 다음과 같은 프로세스 모델을 고려한다.

Definition 1 (Process Model)

하나의 프로세스를 P라고 할 때, 이는 단위업무의 집합 A와 단위업무 간의 선후관계를 표현하는 링크 집합 L로 표현된다. 즉, $P = (A, L)$이며, 이 때

- $A = \{a_i \mid i = 1, 2, \cdots, I\}$, I는 단위업무의 수
- $L = \{ (a_i, a_j) \mid a_i \in A, a_j \in A,$ 그리고 $i \neq j\}$

이다.

위의 정의에서 L의 원소 (a_i, a_j)는 j-번째 단위업무를 수행하기 위해서는 i-번째 단위업무의 수행이 완료되어 있어야 함을 의미한다. 임의의 서로 다른 두 단위업무 사이에는 다음과 같은 경로가 존재할 수 있다.

Definition 2 (Path)

임의의 두 단위업무 a_i, a_j $(i \neq j)$가 하나 이상의 링크로 연결될 때, 연결된 단위업무들의 집합을 경로라고 하며, 두 단위업무 사이의 경로는 두 개 이상 존재할 수 있다. 따라서, 이들 경로 중 p-번째 경로를 r_p라고 하며, 다음과 같이 표현한다.

$$r_p = \{a_i, a_k, a_{k+1}, \cdots, a_{k+n}, a_j\}$$
$$\text{단, } (a_i, a_k), (a_k, a_{k+1}), \cdots, (a_{k+n}, a_j) \in L$$

즉, 경로는 하나의 단위업무가 또 다른 단위업무로 하나 이상의 링크를 통해 연결되어 있을 때, 이 링크로 연결된 단위업무의 집합을 의미한다.

일반적으로, 프로세스 모델은 정의한 바와 같이 AND 분기, OR 분기, 서브프로세스 등의 구조적 특성에 의해 그 복잡도가 결정된다[Workflow1999]. AND 구조는 모든 경로를 동시에 진행하는 구조인 반면, OR 구조에서는 분기된 여러 경로 중 하나 이상만 수행하면 된다. 즉, AND 구조는 모든 경로를 완료해야 전체 구조를 완료했다고 하고, 하나의 경로라도 완료하지 못하면 전체 구조를 완료하지 못했다고 한다. 반면에 OR 구조는 하나의 경로라도 완료하면 전체 구조를 완료했다고 하며, 모든 경로를 다 완료할 수 없을 때 전체 구조를 완료할 수 없다고 말한다. OR 구조는 분기의 의미에 따라 다시 몇 가지 종류로 나눌 수 있는데[Bae2004], 본 논문에서 다룰 OR 구조의 유형은 NOR (Normal OR), POR (Priority OR), COR (Conditional OR)이다. NOR 구조는 분기 단위업무에서 다수의 경로가 동시에 발생하지만 어떠한 경로든 가장 먼저 병합 단위업무 직전까지 수행을 완료하면 진행 중인 다른 경로를 무시하고 다음 단계의 단위업무를 수행하는 구조이다. POR 구조는 경로에 우선순위를 주어 가장 높은 순위를 갖는 경로를 우선 수행하는 구조이다. 수행중인 경로가 도중에 실패하면 다음 순위의 경로를 시도하고, 반대로 성공적으로 처리되면 나머지 경로를 무시하고 병합 단위업무를 수행한다. COR 구조는 각 경로에 조건이 설정되어 있고, 조건을 만족하는 경로를 진행시킨다. 즉, 조건에 따라 수행되는 경로가 달라진다.

프로세스 모델에서, 서브프로세스는 하나의 독립된 프로세스이면서 상위 프로세스에서는 하나의 단위업무로 취급될 수 있는 특별한 구조이다[Workflow1999]. 실제로 프로세스를 설계할 때, 서브 프로세스의 개념은 매우 유용하게 활용된다. 예를 들면, 업무절차를 설계할 때에 필요한 특정 부분이 이미 독립된 프로세스로 존재한다면, 다시 설계하지 않고, 기존 프로세스를 재사용할 수 있다. 이 때, 기존 프로세스는 설계되고 있는 프로세스에서 서브프로세스로 표현이 된다. 이처럼 서브프로세스는 프로세스를 디지털화하여 관리하는 정보시스템에서 아주 유용한 의미를 지닌다. 기업들은 앞으로 BPO (Business Process Outsourcing), 웹서비스 등을 통해 해당 분야의 Best Practice를 도입하여 경쟁력을 강화해나갈 것이다. 또한, 타 기업과의 협력관계를 구축해나가기 위한 방법으로 서로 다른 기업의 프로세스를 연계, 통합할 필요성이 증대되고 있다[Hammer2001]. 이러한 필요성은 프로세스 구조적인 관점에서 서브프로세스 구조에 의해 충족될 수 있다.

2.3 SCM

1980년대에 접어들면서, 기업들은 새로운 제조 기술과 전략을 도입함으로써 비용을 절감하고 시장에서의 경쟁력을 강화하였다. 그러나, 1990년대에 들어서면서, 많은 기업들은 이와 같은 개별 기업단위의 비용 절감 노력이 한계에 달하였음을 인지하게 되었으며, 글로벌 시장의 경쟁 심화, 제품 수명주기

의 단축, 고객 요구 수준과 같은 시장 환경의 변화에 따라, 공급 사슬 범위의 효과적인 관리의 필요성을 느끼게 되었다. 공급 사슬은 공급자, 제조업체, 창고, 분배센터, 소매점 및 이들을 통해 움직이는 자재, 공정 중 재고, 완성품을 포함한다. 다수의 기업이 관여하는 이러한 공급 사슬에서는 비용을 줄이고 서비스 수준을 높이기 위해서 각 단계 사이에서 일어나는 상호작용을 고려하고 이들에 대해 조정이 수행되어야만 한다. 이러한 공급 사슬 전반의 효율적인 관리를 목표로 하는 활동을 공급 사슬 관리(SCM) 라 한다. 공급 사슬 관리는 "정확한 양의 상품이, 정확한 장소로, 정확한 시간에 생산되고 분배되며, 요구되는 서비스 수준을 충족하면서 시스템 비용을 최소화하기 위해, 공급자, 제조업체, 창고, 설비들을 효과적으로 통합하기 위한 일련의 접근 방법[배2004]"으로 정의된다.

공급 사슬 관리는 앞서 정의한 바와 같이 시장의 요구를 완벽하게 충족시키기 위한 방법으로서 타 기업과의 협력의 필요성이 제기되어 구축된다[배2004]. 즉, 공급 사슬 관리는 그 자체가 목적이 아니라, 시장 상황에 보다 효과적으로 대응하기 위한 방법인 것이다. 외형적으로 공급 사슬 관리가 이루어진다고 하더라도, 이를 통해 시장 경쟁력을 강화하고, 고객의 요구사항을 충족시키지 못한다면 존재 의미가 없을 것이다. 따라서, 공급 사슬 관리를 "효율적으로" 운영하는 것이 매우 중요하다. 공급 사슬 관리에서는 여러 기업이 참여하기 때문에, 한 기업의 효율성이 높다고 하더라도, 이것이 공급 사슬 관리 전체의 효율성 향상으로 직결되지는 않는다. 참여 기업은 전체 공급 사슬이 효율적으로 진행될 수 있도록 개별 업무를 수행해야 한다. 이에 본 논문에서는 각 참여 주체에게 현 시점에서의 공급 사슬 관리 진행 관련 정보를 제공하고, 이를 토대로 한 업무수행방법을 개발하여 제안한다.

3. 우선순위규칙(Dispatching Rule)

BPM 시스템 내의 프로세스 실행 엔진은 정의된 프로세스 모델을 해석하여 적절한 시점에 업무를 사용자에게 할당하고, 사용자는 할당된 업무를 처리한다[Rhee2004]. 이 업무 할당과 처리는 보통 미리 정해진 규칙에 따라 이루어진다. 업무 할당에서 중요한 것은 업무 처리 적임자의 선택이다. 보통 이 적임자는 정의시 단계에서 해당 단위업무의 성격에 맞는 규칙을 기반으로 실행시에 지정된다[Rhee2004]. 그리고, 할당된 업무를 수행하는 업무담당자의 관점에서는 BPM 내에 여러 프로세스 모델이 존재하고 각 프로세스 모델을 따르는 다수의 인스턴스가 동시에 진행되는 것이 보통이다. 따라서, 개별 사용자에게 한 시점에 여러 업무가 할당되어 있는 것이 일반적이다. 개별 사용자에 할당된 업무집합을 업무목록(*worklist*)이라 한다. 사용자는 업무의 기대소요시간 및 업무기한 등을 고려하여 업무항목집합의 업무에 대하여 우선순위를 결정하고 업무를 처리하게 되는데, 이를 우선순위규칙(*Dispatching rule*)이라고 한다.

우선순위 규칙은 원래 제조 공정의 스케줄링 및 할당 문제의 해법으로 연구되어 왔다[Baker1974]. FIFO (First In First Out), SPT (Shortest Processing Time), Random Rule 등 다양한 규칙이 해당 분야에서 응용되어 활용되어 왔고, 그 적용 효과에 대한 연구 결과도 매우 많다[Park1997]. BPM 프로세스 모델에 우선순위 규칙을 적용한 연구는 많지 않으며, 프로세스의 효율을 높이기 위해 우선순위 규칙을 적용한 연구[Rhee2004]가 존재한다. 이 연구에서는 여유시간을 각 업무담당자의 단위업무에 제공하여 가장 작은 여유시간 값을 갖는 단위업무를 우선적으로 처리하는 LST (Least Slack Time) Rule을 적용하고 있다. [Rhee2004]에 따르면 LST를 적용한 BPM 프로세스 모델이 AND 구조를 많이 포함할수록, 프로세스 인스턴스 발생빈도가 높을수록 전체 프로세스 효율성의 개선효과가 점차 증가하는 경향이 있으며, 단위업무의 대기건수도 감소하는 것으로 알려져 있다.

본 연구에서는 효율적인 공급 사슬 관리를 위한 업무수행방법으로서 이러한 LST 규칙을 사용한다. LST 규칙을 사용하면 주경로 및 여유시간을 통해 전체 SCM 관점에서 각 업체들의 진행상황을 쉽게 파악할 수 있다. 일반적으로, 공급 사슬에 참여하는 각 기업들은 자사 내부 진행 상황에 초점을 두기 때문에, 공급 사슬 전체의 효율성을 고려하기 어렵다. 따라서, 각 참여 주체는 특정 시점에서 제공되는 여유시간 정보를 토대로 업무를 수행하기만 하면, 공급 사슬 전체의 효율성을 향상시키는 효과를 기대할 수 있다. 또한, [Rhee2004]의 결과에서와 같이 공급사슬이 여러 업무들이 동시에 진행해야 하는 상황이 빈번하거나, 시장의 수요가 많을 경우 LST 규칙의 효과는 더욱 커질 것이다. 이처럼, LST 규칙을 적용함으로써 공급 사슬의 효율성 향상을 기대할 수 있으며, LST Rule을 SCM 프로세스에 적용하기 위해서는 몇 가지 사항을 추가적으로 고려해야 한다. 이에 대한 자세한 사항은 4장에서 논의하기로 한다.

4. 효과적인 SCM 프로세스 실행을 위한 고려사항

본 장에서는 일반적인 SCM을 BPM이 실행 가능한 프로세스 모델로 표현하고, LST 규칙을 적용하여 이를 효율적으로 실행하기 위해 고려해야 할 사항을 논의한다. 먼저, SCM을 BPM에서 실행 가능한 프로세스 모델로 표현하기 위해 필요한 내용을 설명한다. 그리고, BPM으로 표현된 SCM 프로세스 모델에 [Rhee2004]에서 제안한 LST Rule을 적용하는데 필요한 고려사항을 설명한다.

4.1 BPM 상에서 구현한 SCM 프로세스

효과적인 SCM을 시스템 상에서 구현하기 위해서는 서로 다른 업체들 사이에 존재하는 이기종 시스템 간의 통합을 통해 공급 사슬 전반에 걸쳐 재고 및 최종 수요 정보의 실시간 공유가 필요하다. 또한, 각 업체들의 프로세스를 연결하여 전체 프로세스 관점

에서 실시간 의사결정 및 통합적 운영이 필요하다. 최근 발전하고 있는 BPM은 이러한 공급사슬의 통합적 운영을 구현하는 데에 적합한 시스템이 될 수 있다. BPM은 정보공유를 원활하게 해주며 공급 사슬 프로세스 내의 각 활동을 모니터링하여 운영상의 의사결정을 위한 정보를 실시간으로 제공하는 것을 목표로 한다.

한편, 이러한 과정에서 상호작용이 원활하게 이루어지기 위해서는 프로세스 정보의 상호 교환이 필요하게 되는데, 각 참여주체들이 이기종 시스템을 채용하는 경우 프로세스 모델에 대한 표준적인 언어가 필요하다. 본 연구에서는 이러한 프로세스의 표준적인 모델링을 위하여 BPEL (Business Process Execution Language)을 활용한다[배2004]. BPEL은 원래 웹서비스 기반의 비즈니스 프로세스 기술언어이지만, BPM에서 사용되는 시스템 의존적인 PDL (Process Definition Language)에 비해 의미적인 표현력이 떨어지지 않은 것으로 알려져 있다[Wohe2002]. BPM에서 다루는 프로세스 모델을 BPEL로 변환하는 주요 방법은 간단하게 표1와 같이 정리할 수 있다. 본 논문의 주요 목적은 프로세스 관련 언어 및 SCM 프로세스의 모델링 자체를 논하는 것이 아니므로, 자세한 내용은 [Bae2004][배2004]를 참고하기 바란다.

표 1. BPEL 모델 변환

Pattern	Structured	(Structured) BPEL
Serial	Sequence	<sequence> … </sequence>
Conditional Split	OR Structure	<switch> <case condition="조건식1">…</case> <case condition="조건식2">…</case> </switch>
Parallel	AND Structure	<flow>… </flow>

전술한 바와 같이 SCM 프로세스는 BPEL을 통해 모델링 될 수 있으며, 이 모델은 BPM 상에서 효과적인 운영이 기대된다. 따라서, SCM 프로세스는 BPM 프로세스 모델의 한 유형으로 간주할 수 있으며, LST 규칙의 적용을 위해 BPM 프로세스 모델의 구조와 속성을 고려할 필요가 있다.

4.2 SCM 프로세스에 적용한 LST 규칙

LST 규칙은 복수 개의 프로세스에 다양한 업무담당자들이 참여하는 상황에 적용된 규칙이다. LST 규칙을 적용하기 위한 첫 번째 단계는 업무의 여유시간 계산이다. 이를 위해 [Rhee2004]에서는 PERT/CPM을 활용하였다. 주경로 및 여유시간 값이 계산되면, 어떤 경로가 집중 관리를 요하는지 또는 각 활동이 얼마나 시간적 여유가 있는지를 토대로 사용자가 업무처리순서를 결정할 수 있다. 이러한 점은 SCM 프로세스의 효율성 향상을 위해서도 매우 중요하다.

그러나, SCM 프로세스에 LST 규칙을 적용하는 데에는 몇 가지 제약이 있다. 이는 LST 규칙이 고려하는 프로세스는 한 기업 내에서 운영되는 단일 프로세스인 반면, SCM 프로세스는 서로 다른 기업 간 프로세스의 통합을 전제로 하기 때문이다. LST 규칙을 적용하기 위해서는 각 단위업무의 예상소요시간 및 실제소요시간의 정보를 통해 주경로와 여유시간을 계산해야 하는데, 이를 위해서는 프로세스 모델의 "구조"를 고려해야 한다. 만약, SCM 프로세스라고 해도 단일 프로세스 모델로 서로 다른 기업 간 공급 사슬을 표현할 수 있다면, 기존 LST Rule을 바로 적용할 수 있을 것이다. 그러나, 웹서비스 또는 프로세스 인터페이스를 통해 별도의 프로세스를 호출, 통합해야 한다면 프로세스 구조적인 관점에서는 앞서 설명한 바 있는 서브프로세스로 표현해야 한다. 이런 경우, 서브프로세스를 포함하지 않은 단일 프로세스만을 고려한 기존 LST 규칙을 바로 적용하기 어렵다. 그림 1은 웹서비스 등 서로 다른 프로세스 간의 통합을 서브프로세스 구조를 이용하여 나타낸 것이다.

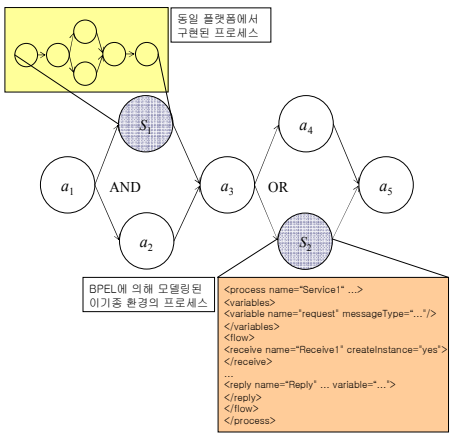

그림 1. 프로세스 통합에 활용된 서브프로세스

글로벌 SCM 프로세스 관점에서 서브프로세스는 하나의 단위업무로 간주될 수 있다. 그러므로, 글로벌 프로세스 내에 존재하는 서브프로세스의 기대소요시간 및 예상소요시간을 단일 단위업무의 값처럼 단일 값으로 계산할 수 있다면, 이를 토대로 서브프로세스 내 단위업무를 포함한 각 단위업무의 여유시간을 계산할 수 있다.

5. SCM 프로세스를 위한 LST 규칙

프로세스 모델이 서브프로세스를 포함하지 않는 경우에는 기존 LST Rule에서 사용하는 여유시간 계산 방식을 그대로 적용할 수 있다. 따라서, 본 절에서는

기존 LST Rule의 방식을 간략하게 설명한 뒤, 서브프로세스를 포함하는 프로세스 모델에 대해 논의한다.

5.1 기존 LST 규칙의 요약

기존 LST Rule에서는 여유시간을 정의 시와 실행 시로 나누어 계산한다. 먼저, 정의시에 각 단위업무의 기대소요시간을 토대로 전체 프로세스의 주경로를 정의하고, 각 단위업무의 여유시간을 계산한다. 프로세스 모델이 AND 구조만 포함하고 있을 경우 PERT/CPM의 시간 계산 방법을 그대로 적용할 수 있다[Rhee2004]. 그러나, OR 구조를 포함하고 있을 경우, PERT/CPM 방법을 그대로 적용할 수 없다. 한편, OR 구조는 실행 시에 하나의 단위업무만을 수행하게 되므로, 하나의 단위업무로 간주될 수 있다. 따라서, OR 구조의 기대소요시간을 하나의 대표값으로 계산한 뒤, PERT/CPM 방법을 통해 계산하면 된다. 각 OR 구조 유형에 따른 대표값 계산방식의 자세한 내용은 [Rhee2004]을 참고하면 된다.
한편, 실행시에는 특정 단위업무가 완료된 시점에 주경로와 여유시간을 재계산할 필요가 있다. 이는 실제 단위업무 수행시간이 기대소요시간과 다를 수 있고, 또 OR 구조의 경우 실제 수행되는 경로가 프로세스 수행 과정에서 결정되기 때문이다. 재계산 알고리즘은 PERT/CPM의 방법을 그대로 사용할 수 있다.

5.2 중첩 프로세스 구조의 처리

서브프로세스를 포함한 전체 프로세스에서 서브프로세스는 하나의 단위업무로서 간주된다. 따라서, 전체 프로세스에서 여유시간을 계산하기 위해서는 보통 단위업무와 마찬가지로 서브프로세스를 대표하는 하나의 소요시간 값을 도출해야 한다.

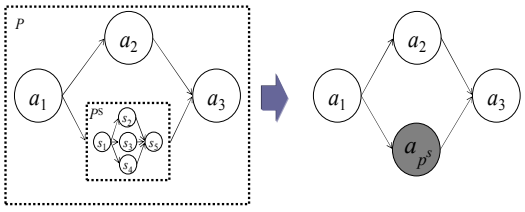

그림 2. 서브프로세스의 대표단위업무

그림 2의 좌측 프로세스에서 서브프로세스 P^S는 단위업무 s_1과 병합 단위업무 s_5 사이에 세 개의 경로 $r_1 = \{s_1, s_2, s_5\}$, $r_2 = \{s_1, s_3, s_5\}$, $r_3 = \{s_1, s_4, s_5\}$를 가진다. 이 서브프로세스 P^S를 하나의 단위업무 a_{p^s}로 표현하여 '대표단위업무'라 하고, 이 단위업무에 대해 기대소요시간을 산출한다. a_{p^s}이 기대소요시간은 서브프로세스의 주경로의 기대소요시간과 같다. 왜냐하면, 서브프로세스의 완료시간은 주경로가 완료되는 시점이기 때문이다. ET_{a_i}와 ET_i를 각각 단위업무 a_i의 기대소요시간 및 경로 r_i의 기대소요시간이라고 하면 a_{p^s}의 기대소요시간은 다음과 같다.

$$ET_{a_{p^s}} = Max\{ET_1, ET_2, ET_3\} \quad \text{식(1)}$$

이를 n개의 경로를 가지는 서브프로세스 P^X로 일반화하면 다음과 같다.

$$ET_{a_{p^x}} = Max\{ET_i \mid 1 \le i \le n,\ i\text{는 자연수}\} \quad \text{식(2)}$$

서브프로세스의 기대소요시간이 계산되면, PERT/CPM의 알고리즘을 통해 글로벌 프로세스의 주경로와 여유시간을 계산할 수 있다. 서브프로세스의 여유시간 또한 대표단위업무의 여유시간으로서 계산된다. 다시 대표단위업무의 여유시간은 서브프로세스 내 단위업무의 여유시간 산출에 활용된다. 프로세스 P의 단위업무 a_i의 여유시간을 (ST_{a_i}, P)이라고 하면, 그림 2에서 전체 프로세스 P의 관점에서 P^S 내 각 단위업무 여유시간은 다음과 같이 계산된다. 위의 예제에서는 P^S 내에 5개의 단위업무가 포함되어 있으므로, n은 5가 된다.

$$(ST_{s_i}, P) = ET_{a_{P^s}} + (ST_{s_i}, P^S) \quad i = 1, 2, \cdots, n \quad \text{식(3)}$$

식 (3)은 PERT/CPM의 계산방식을 적절하게 변경한 것으로 P^S의 단위업무가 전체 프로세스 레벨에 위치한 단위업무라고 가정하여 계산한 여유시간 값과 일치하게 된다. 한편, 서브프로세스를 포함한 프로세스에서도 주경로와 여유시간을 재계산할 필요가 있다. 실행시에 서브프로세스 내 시간 정보도 변화하기 때문이다. 이 같은 재계산의 필요성이 발생할 때마다 주경로와 여유시간을 갱신해야 한다. 재계산은 기대소요시간을 기준으로 이루어지는 여유시간 계산방식과 동일하며, 완료된 단위업무에 대해서 기대소요시간 대신 실제완료시간을 대입하면 된다. 갱신 시점을 감지하여 전체 프로세스의 여유시간 정보를 재계산하는 알고리즘은 [Kim2002]을 통해 참고하면 된다.

5.3 BPEL을 이용한 여유시간 정보의 활용

본 논문에서는 공급사슬관리 프로세스처럼 서로 다른 환경에서 구축된 프로세스 간의 통합을 구현하기 위해 BPEL을 사용할 것을 제안하였다. SCM 프로세스에 LST 규칙을 적용하려면, 서로 다른 프로세스 간에 여유시간 정보를 전달할 수 있어야 한다. 예를 들어, 서브프로세스를 포함한 전체 프로세스에서 여유시간을 계산하려면, 다른 환경에서 실행되는 서브프로세스의 여유시간을 전체 프로세스에 가져올 수 있어야 하며, 반대의 상황도 가능해야 한다. 본 논문에서는 하나의 프로세스 관점에서 여유시간 정보가 지속적으로 관리될 수 있도록 하기 위해 BPEL의

프로세스 변수 태그에서 여유시간을 정의한다.
이기종 환경에서 구현한 프로세스를 서브프로세스로 포함하고 있는 프로세스에서 크게 두 가지 정보를 프로세스 변수로서 정의한다.

1) pv_slacktime

정의	특정 프로세스 내 각 단위업무의 여유시간
타입	문자 타입
형식	단위업무1ID:여유시간값1/단위업무2ID:여유시간값2/단위업무3ID:여유시간값3/…
예	101:0.39/102:0/103:1.245

2) pv_overall_st

정의	전체 프로세스 관점에서 각 단위업무의 여유시간 (식(3)에 의해 계산됨)
타입	문자 타입
형식	단위업무1ID:여유시간값1/단위업무2ID:여유시간값2/단위업무3ID:여유시간값3/ …
예	101:0/102:0/103:0

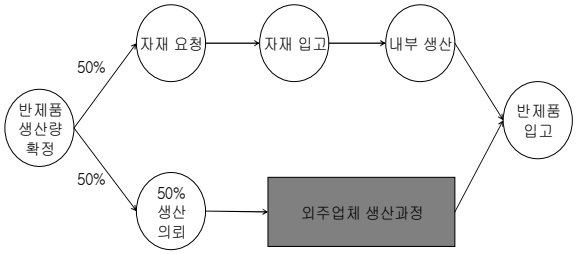

그림 3. 프로세스 변수 활용 예제 프로세스

본 논문에서 정의한 두 가지 프로세스 변수의 활용예를 예제 프로세스를 통해 살펴보자. 위의 예제 프로세스는 반제품 생산절차를 아주 간단하게 표현한 것이다. 위 프로세스에서 생산해야 할 반제품의 50%은 외부업체에 의해 처리되는데, 이 과정은 사각형으로 표기된 부분에 해당되고, 서브프로세스로 표현되어 하나의 단위업무로 인식된다. 서브프로세스를 여유시간 정보의 전송을 고려하여 BPEL로 정의할 수 있으며, 이 때 여유시간은 프로세스 변수로 설정할 수 있다.
예제 프로세스에서 전체 프로세스를 p1, 서브프로세스에 해당하는 외주업체 생산과정을 p2라고 하자. 또한, 특정 시점에서 p1의 pv_slacktime은 5.2절에서 논의한 계산식을 토대로 아래와 같다고 하자. 이 때, p2는 하나의 단위업무로 인식하여 단위업무 id 110이라고 한다.
pv_slacktime= 101:1.12/102:0/103:1.245/…/110:1.245

이 때, p2가 세 개의 단위업무로 구성되어 있다고 가정하고, p1에서 가정한 시점과 동일 시점에서 pv_slacktime을 아래와 같다고 한다.
pv_slacktime = 101:0/102:0/103:0.24

이렇게 하면, 전체 프로세스 관점에서 p2 내의 단위업무 여유시간은 다음과 같이 계산된다.
pv_overall_st
= 101:**1.245**+ 0/102:**1.245**+ 0/103:**1.245**+ 0.24
= 101:1.245/102:1.245/103:1.269
이 된다.

LST 규칙은 pv_overall_st 값을 통해 적용되기 때문에, 전체 프로세스와 서브 프로세스 간 pv_slacktime 정보가 정확하게 전달되어야 한다. 따라서, 두 프로세스 변수는 특정 단위업무가 완료되는 시점에서 항상 새롭게 계산된 값으로 업데이트되어야 한다.

지금까지 BPM 상에 구현된 공급사슬관리 프로세스 변수로 정의하여 모델링한 예제 프로세스의 BPEL 정의서이다.

```
<process name="supplychain" targetNamespace=
"http://abc.com/supplychain" suppressJoinFailure="yes"
xmlns:tns="http://abc.com/supplychain"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/
business-process/"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:factory="http://abc.com/factory"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <partnerLinks>
  <partnerLink name="client" partnerLinkType=
   "tns:supplychain" myRole="supplychainProvider"
   partnerRole="supplychainRequester"/>
  <partnerLink name="factoryPT"
   partnerLinkType="factory:factory"
   partnerRole="factoryProvider"
   myRole="factoryRequester"/>
 </partnerLinks>
 <variables>
  <variable name="input"
   messageType="tns:supplychainRequestMessage"/>
  <variable name="output"
   messageType="tns:supplychainResponseMessage"/>
  <variable name="pv_slacktime" type="xsd:string"/>
  <variable name="pv_overall_st" type="xsd:string"/>
 </variables>
 <sequence name="main">
  <receive name="주문접수" partnerLink="client"
   portType="tns:supplychain" operation="initiate"
   variable="input" createInstance="yes"/>
  <flow name="flow-1">
   <sequence>
    <invoke name="보유분 발송"/>
    <flow name="flow-2">
     <sequence>
      <receive createInstance="no" name="
       물품도착 및 인수"/>
      <invoke name="이월주문처리"/>
     </sequence>
     <sequence>
      <invoke name="재고량 갱신"/>
      <invoke name="재주문 여부판단"/>
      <switch name="switch-1">
       <case>
        <empty name="empty-1"/>
       </case>
       <otherwise>
        <sequence>
         <invoke name="생산의뢰" partnerLink=
          "factoryPT" portType="factory:factory"
          operation="initiate"/>
         <receive createInstance="no" name=
          "물품도착" partnerLink="factoryPT"
          portType="factory:factoryCallback"
          operation="onResult"/>
         <invoke name="물품인수"/>
        </sequence>
       </otherwise>
      </switch>
     </sequence>
    </flow>
   </sequence>
   <sequence>
    <invoke name="부족분 생산의뢰"/>
   </sequence>
  </flow>
  <invoke name="재고계산"/>
  <invoke name="callbackClient" partnerLink="client"
   portType="tns:supplychainCallback" operation="onResult"
   inputVariable="output"/>
 </sequence>
</process>
```

프로세스 변수 정의

그림 5. BPEL을 이용한 SCM 프로세스 정의

프로세스에 LST 규칙을 적용하여 효율적으로 운영하는 방법을 논의하였다. 본 절에서 제안한 방법을 이용하면, BPEL을 통해 기업 간 프로세스의 여유시간 정보를 교환할 수 있으며, 이를 통해 LST 규칙을 적용할 수 있다. 따라서, 다음 장에서는 본 논문에서 제안한 BPEL을 이용한 모델링 방식을 토대로 간단한 프로토타입을 구현함으로써, LST 규칙의 활용 가능성을 보인다.

6. 프로토타입의 구현

본 장에서는 제안한 방법을 프로토타입으로 구현한다. 이를 위해 아래 예제 프로세스를 BPEL을 통해 모델링하고, 그 결과를 보인다.

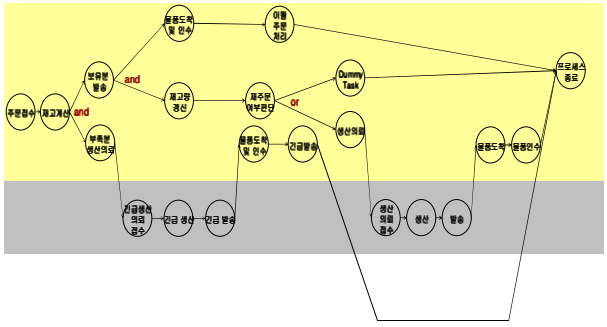

그림 4. SCM 프로세스 예제

예제 프로세스에서 회색으로 표기된 단위업무들의 집합을 하나의 서브프로세스로 정의한다. 따라서, 예제 프로세스에서는 2개의 서브프로세스를 포함하게 된다. 아래 그림 5는 두 개의 여유시간 정보를 프로

7. 결론

본 논문에서는 이기종 환경에서 구현된 서로 다른 프로세스 간 상호운용이 요구되는 SCM을 효율적으로 관리하는 방안에 대해 논의하였다. LST 규칙은 BPM 프로세스 모델을 효율적으로 관리하는 방법 중 하나로서, SCM 프로세스에서도 제안한 방법을 통해 BPEL을 이용하여 SCM 프로세스를 효율적으로 관리하도록 한다. LST 규칙을 통해 SCM 전체 프로세스 관점에서 단위업무의 처리현황을 자세하게 확인할 수 있으며, 단위업무 처리 진척도에 따라 업무처리순서를 제공할 수 있는 장점이 있다.

참고문헌

[배2004] 배혜림, 서용원, 최용선, 장진영, “BPM을 이용한 웹서비스 기반의 SCM 프로세스 실행” 한국전자거래학회지, 9권, 4호, 2004.

[Bae2004] Bae, J., H. Bae, S. Kang, and Y. Kim, “Automatic Control of Workflow Processes using ECA Rules,” IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 8, 2004.

[Baker1974] Baker, K.R. “ Introduction to Sequencing and Scheduling, ” Wiley, New York, 1974.

[Hammer2001] Hammer, M. “Agenda: What Every Business Must Do to Dominate the Decade,” Random House, 2001.

[Kim2000] Kim, Y., Kang, S., Kim, D., Bae, J. and Ju, K. “ WW-Flow: Web-based Workflow Management with Runtime Encapsulation, ” IEEE Internet Computing, Vol. 4, No. 3, pp. 55-64, 2000.

[Kim2002] Kim, Y. and S. Rhee, Algorithm of Real Time Computation of Slack Time for Workflow Processes, SNU Information Systems Lab. Technical Report, SNUIS-TC-1017. 2002.

[Park1997] Park, S. C., N. Raman, and M. J. Shaw, "Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach," IEEE Transactions on Robotics and Automation, Vol. 13, No. 4, pp. 486-502, 1997.

[Rhee2004] Rhee, S.-H. H. Bae, Y. Kim, "A Dispatching Rule for Efficient Workflow," Concurrent Engineering – Research and Applications, Vol. 12, No. 4, 2004.

[Rhee2005] Rhee, S. -H. et al, "Process-Oriented Development of Job Manual System," Lecture Notes in Computer Science, Vol. 3482, pp. 1259-1268, 2005.

[Smith2003] Smith, H. and P. Fingar, "Business Process Management – The Third Wave," Meghan-Kiffer Press, 2003.

[Wohed2002] Wohed, P., W.M.P. Aalst, M. Dumas, and A. H. M. Hofstede, Pattern Based Analysis of BPEL4WS, Technical Report FIT-TR-2002-4, QUT

[Workflow1999] Workflow Management Coalition: Process Definition Interchange, Document No. WfMC-TC-1016-P, 1999, http://www.wfmc.org.