협업 비즈니스 프로세스의 서비스 품질(QoS)에 관한 연구

오제연[1], 조남욱[2], 김훈태[3], 민윤홍[1], 강석호[1]
[1]서울대학교 산업공학과, [2]서울산업대학교 산업정보시스템공학과, [3]대진대학교 산업시스템공학과

Abstract

오늘날 세계화된 시장 환경은 기업간 협업 프로세스의 동적 구성을 요구하고 있으며, 웹 서비스(Web services)와 비즈니스 프로세스 관리(Business Process Management) 기술의 발전이 이의 구현을 뒷받침하고 있다. 임의의 기업과의 느슨한 구성(loosely-coupled)을 통한 동적 협업이 활성화됨에 따라, 프로세스의 서비스 품질 (Quality of Service)이 중요한 문제로 대두되고 있는데, 특히 단순한 단위 웹 서비스가 아닌 장기 실행 액티비티 (Long-termed activity)를 포함하는 협업 프로세스에서는 제한된 시간 내에 프로세스의 정상적 수행을 보장하는 것이 핵심적 요구사항이라 할 수 있다. 본 연구에서는 프로세스 수행 계획을 동적으로 생성, 관리함으로써 프로세스의 각 인스턴스들의 완료와 납기 준수를 보장할 수 있는 방법론을 제시한다. 본 방법론은 프로세스 인스턴스의 진행 과정에 따라 중복(redundancy) 계획을 동적으로 수정함으로써 각 액티비티의 수행 시간과 신뢰성(reliability)을 독립적으로 관리한다. 본 방법론의 최적화 모델은 NP-hard로 증명되었으며, 본 연구에서는 최적화 모델을 위한 휴리스틱 알고리즘을 제시하고, 이를 실제 최적해와 비교하는 실험을 행하였다. 본 연구를 통해 보다 복잡한 기업간 협업 환경에서의 수행 보장과 장애내감성 실현이 가능해질 것으로 기대된다.

1. 서론

오늘날 웹서비스와 비즈니스 프로세스 관리 기술이 기업 간의 동적 협업을 가능하게 하면서 기업간 협업 프로세스에 대한 연구와 응용이 활발히 이루어지고 있다. 그러나 임의의 기업과의 동적인 프로세스 구성이 실제로 활용되기 위해서는 먼저 해결해야 할 과제들이 존재하는데, 프로세스의 서비스 품질 (Quality of Service, 이하 QoS) 관리가 이에 속한다. 그간 웹서비스의 QoS에 관한 연구들이 활발히 이루어져왔음에도 불구하고, 기업간 협업 프로세스의 QoS를 관리하는 데에는 다음과 같은 어려움들이 따른다. 첫째, 프로세스를 구성하는 액티비티들이 서로 다른 여러 공급자들에 의해 실행될 수 있기 때문에, 전체 프로세스의 QoS를 한 조직에서 관리하기 힘들다. 둘째, 협업 프로세스는 장기 실행 액티비티 (Long-termed activity)를 포함하고 있을 수 있는데, 이 경우의 프로세스의 수행시간과 신뢰성 등의 QoS 평가기준들은 일반적인 웹서비스의 QoS와 다르게 관리되어야 한다.

본 연구에서는 기업간 협업 프로세스의 실행 보장을 위한 프로세스 런타임에서의 동적 실행계획 수립 방안을 제시한다. 제안된 방법론은 프로세스의 납기 내에 성공적 수행을 보장하기 위해 중복생성을 통한 정보시스템의 장애 복구[1] 방법론을 응용하여 프로세스 실행계획을 수립한다. 본 방법론은 납기 내 수행을 보장하는 범위 내에서 비용을 최소화하는 문제로 정형화된다. 해당 문제는 NP-hard이며, 본 연구에서는 이를 위한 휴리스틱 알고리즘을 제시하고, 최적해와의 성능 차이를 비교 실험하였다.

2. 관련 연구

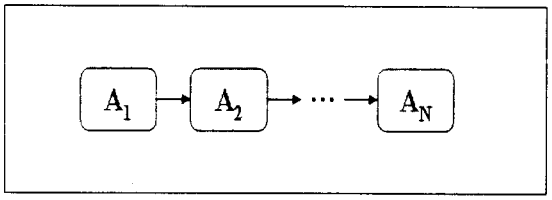
웹서비스 환경에서의 QoS는 다양한 평가요소들[2]과 관련 주제들[3]을 포함하고 있다. 그중 복합 웹서비스의 QoS측정에 대해 많은 연구들이 이루어져 왔다[4][5]. 이는 복합 웹서비스의 QoS측정에 해당 웹서비스를 구성하는 단위 웹서비스들의 QoS 측정 결과를 이용하는 것으로써, 직렬·병렬·조건분기 등의 다양한 제어 흐름에서 수행시간·신뢰성·비용 등의 각 QoS 요소들을 취합하여 전체 QoS를 계산하는 것을 목적으로 한다. 활발한 연구가 이루어지고 있는 또 다른 분야로 QoS에 기반한 웹서비스의 선택 문제가 있다[6][7]. 이는 복합 웹서비스를 구성하는 단위 웹서비스들이 다수의 공급자 후보들을 보유하고 있는 경우 수행시간 최소화·비용 최소화 등 QoS 요구조건을 만족하는 공급자 조합의 선택을 목적으로 한다.

본 연구는 정보시스템의 장애허용성[8]에 대한 연구들도 활용하고 있으나, 서비스 중심 아키텍처(Service Oriented Architecture, 이하 SOA)에서의 장애복구에 적용 가능한 일부분만을 활용한다.

3. 협업 프로세스의 실행 보장을 위한 동적 실행 계획

3.1. 문제정의 및 정형화

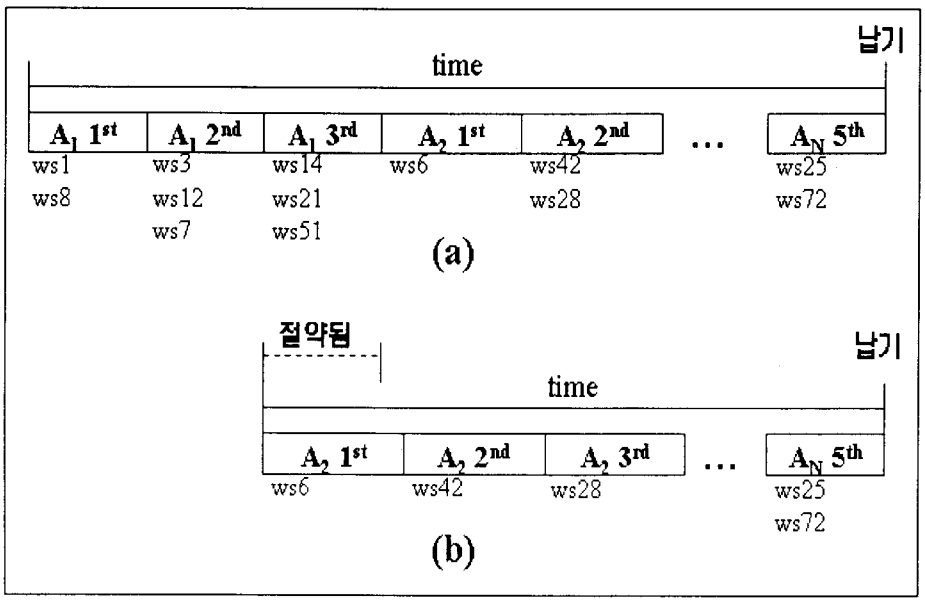
[그림1]과 같이 N개의 액티비티로 구성된 프로세스



[그림 1] N개의 액티비티로 구성된 협업 프로세스

를 가정한다. 각 액티비티들이 웹서비스로 이루어져있다고 가정한다면 프로세스의 실행 보장에 웹서비스 선택 문제를 활용할 수 있다. 이 경우, i번째 액티비티의 신뢰성이 r_i로 측정되었을 때 해당 프로세스의 신뢰성은 Πr_i로 계산되고[4], 사용자가 해당 값을 최대화하도록 서비스 공급자들을 선택할 수 있다. 그러나 선택된 서비스들 중 하나 이상이 실패하는 경우 장애 복구를 위해서는 별도의 방법론을 필요로 하며, 실행 가능한 웹서비스들 중 요구된 신뢰성을 만족시키는 서비스가 존재하지 않을 경우에는 실행 계획이 불가능할 수 있다. 본 연구에서는 먼저 장애 복구를 위해 런타임에 프로세스 실행 결과에 따라 실행전략을 변경함으로써 각 액티비티에 요구되는 신뢰성을 독립적으로 관리하는 방법을 사용한다. 그리고 각 액티비티의 요구된 신뢰성을 만족시키기 위해 액티비티의 인스턴스를 여러 개 수행한다. 발생된 인스턴스들 중 1개의 인스턴스가 완료되면 이후의 액티비티를 진행할 수 있다. 이를 워크플로우 패턴으로 나타내면 Static Partial Join for Multiple Instances Pattern[9]이 된다. 이 방법은 웹서비스의 성공 확률을 보장할 수 있는 간단한 방법이지만, 프로세스에 납기가 추가된다면 이 방법만으로는 신뢰성의 관리에 불충분하다. 납기와 비용을 고려한, 그리고 실행중 발생하는 변화에 대응할 수 있는 최적의 실행전략을 찾기 위해 본 연구에서는 납기에 따라 각 액티비티에 합당한 시간을 할당하고, 할당된 시간 내에 액티비티에 요구된 성공 확률을 만족하는 방법론을 제안한다. [그림2]는 본 방법론의 예시를 나타낸다. [그림2](a)는 프로세스 실행전 결정되는 최적의 실행전략이다. 액티비티 A_1의 신뢰성을 보장하기 위해 8개의 액티비티 인스턴스를 발생시키며, 각 인스턴스는 서로 다른 웹서비스 (ws1, ws8, ws3, ws12, ws7, ws14, ws21, ws51)를 통해 실행된다. 그러나 해당 프로세스 인스턴스가 실행될 때 8개의 서비스들이 모두 동시에 요청되지는 않는다. 8개의 서비스를 3번의 시도에 걸쳐 실행함으로써 우리는 A_1의 수행을 보장하면서 실행 비용을 줄일 수 있는 기회를 얻는다. 예를 들어 A_1의 두 번째 시도에서 서비스 ws3, ws12, ws7중 하나가 성공적으로 수행되었다면, A1을 위한 3번째 시도는 불필요한 것이 되므로 ws14, ws21, ws51은 수행하지 않게 된다. [그림 2](b)가 이 상황을 나타내고 있다. A_1의 3번째 시도의 비용과 시간이 절약되었으므로, 나머지 액티비티들의 수행에 할당되는 시간이 늘어나게 된다. 따라서 A_2의 실행을 위해 ws28의 수행비용이 절약될 수 있는 가능성이 늘어난다.

본 실행 전략을 정형화하기 위해 다음과 같은 가정들을 한다. 먼저 모든 액티비티들을 수행할 수 있는 서비스들은 충분히 많으며, 액티비티 Ai를 실행하기 위한 서비스들은 모두 동일한 가격 c_i, 수행시간 t_i, 성공확률 λ_i $(0 \le \lambda_i < 1)$을 가진다고 가정한



[그림 2] 실행계획의 동적 수정안 예시

[표 1] 프로세스 실행비용 및 수행시간의 기댓값

A_i의 실행비용	$C(A_i)=\sum_{j=1}^{x_i}\left[c_i y_{ij}\cdot\prod_{k=1}^{j-1}(1-\lambda_i)^{y_{ik}}\right]$
A_i의 수행시간	$T(A_i)=\sum_{j=1}^{x_i}\left[jt_i\left[1-(1-\lambda_i)^{y_{ij}}\right]\cdot\prod_{k=1}^{j-1}(1-\lambda)^{y_{ik}}\right]$
목적함수	$\sum_{i=1}^{N}C(A_i)+P\cdot MAX\{0,T(A_i)-Td\}+R\cdot\min\{0,T(A_i)-Td\}$
제약조건	$(1-\lambda_i)^{\sum_{j=1}^{x_i}y_{ij}}\le\varepsilon$, $0<\varepsilon<1$: 매우 작은 임의의 수 $(1-\lambda_i)^{\sum_{j=1}^{x_i}y_{ij}-1}>\varepsilon$ $y_{ij}\ge 1$.

다. 전체 프로세스를 완료해야 하는 기한을 Td라 하고, 기한을 넘긴다면 매일 P의 페널티가, 기한 전에 마칠 경우 매일 R의 보상이 지급된다고 가정한다. 하나의 액티비티 A_i를 수행하기 위해 x_i번의 수행을 할 시간이 주어지고, 각 시도당 요청하는 서비스 수를 y_{ij}로 두면, 액티비티 A_i의 실행을 위한 실행 비용과 수행 시간의 기댓값, 그리고 최종 목적 함수는 [표1]과 같이 계산된다. 해당 문제는 P가 매우 크고 R의 값이 0일때, 그리고 제약조건에서 모든 $\sum y_{ij}$ 값이 $1<\sum y_{ij}\le 2$가 되도록 λ_i와 ε_i의 값이 정해진 경우에 0-1 Knapsack 문제로 유도되어 NP-hard임이 증명된다.

3.2. 휴리스틱

본 연구에서 제안하는 휴리스틱은 각 액티비티의 수행에 소비할 시간의 배정을 계산한 다음 각 액티비티의 서비스 실행 전략의 배정을 따로 계산한다.

먼저 시간이 배정되었을 때의 서비스 실행 전략을 계산하기 위해 하나의 액티비티만으로 이루어진 프로세스가 있다고 가정하면, Td는 모두 한 액티비티의 수행에 사용될 수 있다. 즉 $x_it_i\le Td<(x_i+1)t_i$가 된다. 이 경우, j번째 시도에 의해 발생하는 비용의 기대값은 다음과 같이 계산된다.

$y_{ij}c_i+(1-\lambda_i)^{y_{ij}}(y_{ij}c_i+Pt_i)+P(x_it_i-Td)$, $j\ge x_i$

$y_{ij}c_i+(1-\lambda_i)^{y_{ij}}[y^*_{ij+1}c_i+(x_it_i-Td)(P-R)+Rt_i]+R[(x_i-1)t_i-Td]$, $x_i-1\le j<x_i$

$y_{ij}c_i+(1-\lambda_i)^{y_{ij}}[y^*_{ij+1}c_i+Rt_i]+R[(x_i-2)t_i-Td]$, otherwise.

2, 3번째의 경우 볼록함수이므로 최적값의 계산이 간단하다. 첫번째의 경우 볼록함수는 아니지만 $y>0$인 구간에서 연속이고 ε에 의해 범위가 정해지기 때문에 최소값을 구할 수 있다. 위의 방법을 통해 각 시도별 최적 서비스 수를 계산하는 함수를 getServiceStrategy라 명명하면, 휴리스틱은 다음과 같이 도출된다. 해당 휴리스틱의 복잡도는 $O(n^2)$으로 계산된다.

(1) 각 액티비티에 대해 $q_i = c_i * t_i / \lambda_i$ 를 계산하고 *RemainingTime*:=*Td* 이라 둠.

(2) 각 액티비티에 대해 다음을 계산.

(2-1) 초기배정시간 z_i = *RemainningTime* * q_i / $\sum q_i$

(2-2) 배열 aY_i = getActivityExecutionPlan(A_i)
추정 기대비용 $E(C_i)$, 기대시간 $E(T_i)$을 계산

(2-3) 남은 액티비티들에 배정될 시간을 재계산
RemainingTime := MAX{1, *RemainingTime* - $E(T_i)$}

(3) 초기 최적해 *sOpt* = {aY_1, aY_2, …, aY_N} 와 해당 전략에 따르는 최종 실행비용 계산

(4) 액티비티들을 $E(T_i)/z_i$, 즉 배정된 시간에 대한 기대 소요시간의 비율로 정렬.

(5) (4)에서 정렬된 액티비티들의 순서대로,
(5-1) $z_i'=z_i+[Td-E(T_Opt)]$로 시간 재배정
(5-2) 그에 따른 해와 최종 실행비용 재계산
(5-3) 초기최적해의 실행비용보다 작다면 새로운 최적해로 책정

4. 실험 및 결과

휴리스틱이 계산 시간에서 가지는 이점에 비해 발생하는 해답의 정확도 저하를 평가하기 위해 분기한정법으로 구한 최적해와의 비교 실험을 행하였다. 두 알고리즘은 모두 JAVA로 구현하였으며 2.0GHz AMD Processor에 1GB RAM에서 실행되었다. 실험 변수는 다음과 같이 책정하였다.
(1) 문제 사이즈 : 문제의 사이즈는 액티비티의 개수와 $\sum y_{ij}$ 값의 범위에 의해 결정된다. 본 실험에서는 다음과 같이 책정하였다.
a. 작음 : 액티비티 개수=3, $1\leq\sum y_{ij}\leq3$
b. 중간 : 액티비티 개수=5, $4\leq\sum y_{ij}\leq6$
c. 큼: 액티비티 개수 = 7, $7\leq\sum y_{ij}\leq9$
(2) 납기 제한
a. 느긋함 : $2\sum t\leq Td\leq3\sum t$
b. 급박함 : $0.8\sum t\leq Td\leq1.2\sum t$
(3) 페널티 및 보상 비율
a. 작은 페널티 : P, R≤0.5·(c의 평균)
b. 큰 페널티 : P,R≥2·(c의 평균)

본 실험의 결과는 [그림3], [그림4]와 같이 나타난다. 최적해와 비교하여 4%이하의 정확도 저하를 확인할 수 있다.

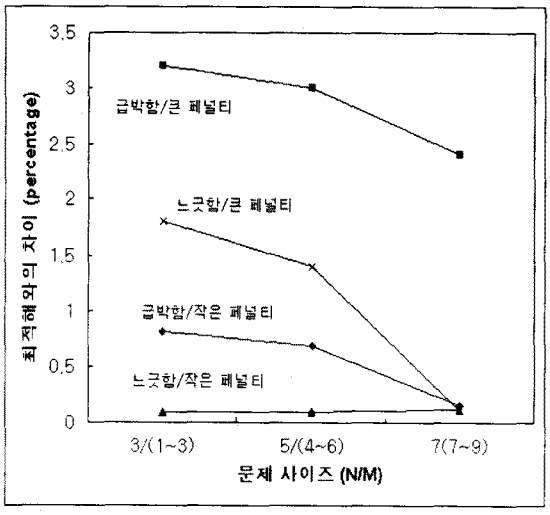
5. 결론

기업간 협업 프로세스의 사용이 증가할수록 QoS의 관리는 보다 필수적인 요소가 된다. 본 연구에서는 협업 프로세스의 실행 보장을 관리하기 위한 동적인 실행계획 수립 방법론을 제안하였으며, NP-hard문제를 위한 휴리스틱을 제시하였다.

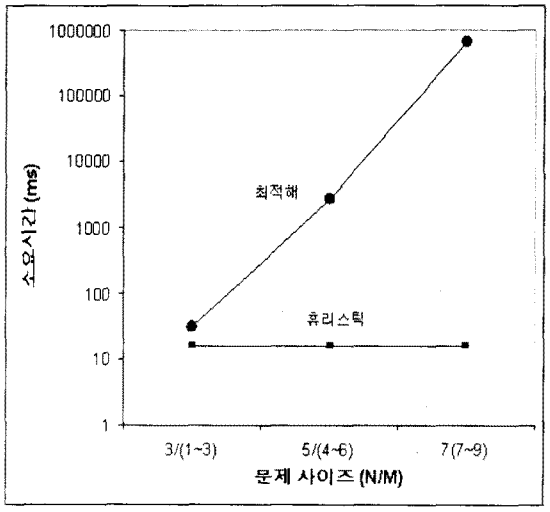
추후 연구 과제로 다음과 같은 문제들이 존재한다. 먼저 본 연구에서는 액티비티를 수행할 수 있는 웹서비스들의 비용과 수행시간, 성공 확률이 동일하다는 가정을 세웠으나, 추후 연구에서는 해당 가정들을 완화할 수 있을 것으로 기대한다. 두 번째로, 다양한 제어 흐름들이 존재하는 경우로 본 방법론을 확장할 수 있을 것으로 기대한다.

참고문헌

[1] Johnson, B. W. (1996) Introduction to the Design and Analysis of Fault-Tolerant Systems. In: Fault-Tolerant Computer System Design, Pradhan, D. K. ed. Fault-Tolerant Computer System Design. Prentice Hall, pp.1-87
[2] Mani, A. and Nagarajan, A. (2002)



[그림 3] 최적해와 휴리스틱의 성능비교



[그림 4] 최적해와 휴리스틱의 시간 비교

Understanding quality of service for Web services,
http://www-128.ibm.com/developerworks/webservices/library/ ws-quality.html
[3] Menascé, D. A. (2002) QoS Issues in Web Services, IEEE Internet Computing, Vol.6, No.6, pp.72-74
[4] Cardoso, J., Sheth, A., Miller, J., Arnold, J. and Kochut, K (2004) Quality of service for workflows and web service processes, Web Semantics:Science, Services and Agents on the World Wide Web, Vol.1, No.3, pp.281-308
[5] Jaeger, M. C., Goldmann, G.R. and Muhl, G. (2004) QoS Aggregation for Web Service Composition using Workflow Patterns, Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International(EDOC'04), pp.149-159
[6] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M. Kalagnanam, J. and Chang, H. (2004) QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering, Vol.30, No.5, pp.311-328
[7] Ardagna, D. and Pernici, B. (2005) Global and Local QoS Constraints Guarantee in Web Service Selection, Proceedings of the IEEE International Conference on Web Services(ICWS'05), pp.805-806
[8] Kopetz, H. and Verissimo, P. (1993) Real Time and Dependability Concepts, In: Mulender, S. ed. Distributed Systems, Workingham: Addison-Wesley, 2nd ed., pp.411-446
[9] Russell, N., Hofstede, A. H. M., Aalst, W. M. P. and Mulyar, N. (2006) Workflow Control-Flow Patterns: A Revised View, BPM Center Report BPM-06-22, BPMcenter.org