creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학박사 학위논문

웹 서비스 기반 협업 프로세스의 동적 구성에 관한 연구

Dynamic Composition for Web Service Based Collaborative Business Processes

2012 년 8 월

서울대학교 대학원

산 업 공 학 과

민 영 빈

국문 초록

기업 구조가 핵심 역량 중심으로 재편되고, 아웃소싱 형태의 협업이 강조됨에 따라 기업간으로 확장된 업무 프로세스를 효과적으로 관리하는 것이 BPM의 주요 기능으로 강조되고 있다. 또한 업무 프로세스의 범위가 기업 외부로 확장됨에 따라, 유연성을 높이는 것이 협업 프로세스의 성공을 확보하는 주요 요인으로 인식되고 있다.

본 논문은 협업 프로세스의 유연성을 높이기 위한 웹 서비스 기반의 협업 프로세스 동적 구성 방법을 다룬다. 대상이 되는 협업 프로세스는 기업 내부의 워크플로우 프로세스와 외부 파트너에 의해 수행되는 웹 서비스 프로세스로 구성된다. 본 연구는 협업 프로세스의 런타임 상황에서 발생하는 정보 미달의 경우를 해결하기 위해 웹 서비스 프로세스를 활용하는 방안을 설명한다.

런타임 프로세스의 유연성을 높이기 위한 기존의 연구들은 상황에 대처할 빌트인 프로세스를 미리 구축해놓고 이를 활용하는 방식을 따르고 있다. 이러한 접근은 전문적인 지식을 가진 프로세스 디자이너에 의존적이고, 모든 런타임 발생 상황을 빌트인 프로세스로 구축해야 하는 현실적으로 불가능하다는 단점을 갖고 있다. 이를 개선하기 위하여 본 연구에서는 인스턴스 상황에 맞춰 프로세스를 동적으로 구성하는 늦은 모델링 방안을 제안하고, 런타임 프로세스의 늦은 모델링에 사용되는 서브 프로세스로 상호운용성을 보장하는 웹 서비스 프로세스를 활용한다.

본 연구는 웹 서비스 프로세스를 이용하여 늦은 모델링을 수행하기 위한 기반인 런타임 프로세스 동적 구성 프레임워크와 그 메카니즘을 제안한다. 그리고 그 과정에서 찾아낸 웹 서비스 선택 최적화 문제를 정의한다. 또한 NP-hard로 증명된 문제를 해결하기 위한 두 가지 휴리스틱을 제시하고 이의 성능을 시험한다.

본 연구의 결과물을 통해, 사용자는 런타임 인스턴스의 정보 미달의 상황을 대응할 수 있는 프로세스를 획득하게 되고, 이는 런타임 프로세스의 유연성을 높이는 기반이 될 것으로 기대한다.

주요어 : BPM, 프로세스 유연성, 런타임 프로세스 동적 구성, 웹 서비스 컴포지션
학 번 : 2007-30166

목 차

1. 서론 ····· 1
1.1. 연구 배경 ····· 1
1.2. 연구 목적 ····· 3
1.3. 연구 구성 ····· 6

2. 관련 연구 ····· 7
2.1. Business Process Management System ····· 8
2.2. 런타임 프로세스의 유연성 ····· 11
2.3. 웹 서비스와 웹 서비스 컴포지션 ····· 16
2.4. 연구 동기 ····· 20
2.4.1. 런타임 프로세스 운영 관점 ····· 20
2.4.2. 웹 서비스 컴포지션 측면 ····· 23

3. 런타임 프로세스 동적 구성 프레임워크 ····· 30
3.1. 웹 서비스 기반 협업 프로세스 모델 ····· 31
3.1.1. 워크플로우 프로세스와 웹 서비스 프로세스의 파트너 독립성 보장 ····· 33
3.1.2. 인스턴스 정보와 웹 서비스 입출력 파라미터의 연결 보장 ····· 35
3.2. 웹 서비스를 이용한 런타임 프로세스 구성 ····· 38
3.2.1. 웹 서비스 네트워크 구축 ····· 41
3.2.2. 프로세스 데이터 오브젝트 매칭 ····· 43
3.2.3. 최적 웹 서비스 선택 및 WS-BPEL 작성 ····· 46

4. 협업 프로세스 동적 구성을 위한 웹 서비스 컴포지션 문제 47
4.1. 네트워크 모형 ····· 48
4.2. 최적화 모형 ····· 51
4.3. 계산 복잡도 ····· 57
4.4. 웹 서비스 선택을 위한 휴리스틱 ····· 59

4.4.1. 후방 검색 (Backward Search) 알고리즘 ····· 60
4.4.2. 유방향 최소신장트리(Directed Minimum Spanning Tree)를 이용한 알고리즘 ····· 65

5. 웹 서비스 네트워크에 따른 협업 프로세스 구성 휴리스틱 성능 실험 ····· 74
5.1. 실험 설계 ····· 75
5.2. 수행 시간 비교 ····· 78
5.3. 비용 비교 ····· 82

6. 결 론 ····· 84
6.1. 요약 ····· 84
6.2. 연구 의의 및 추후 연구 ····· 87

참고문헌 ····· 89

표 목 차

[표 2.1] van der Aalst and Jablonski의 프로세스 변화 판단 기준 ····· 12
[표 5.1] 휴리스틱 알고리즘의 비용 오차율의 평균 ····· 82

그 림 목 차

<그림 1.1> 연구 목적의 도출 ····· 3
<그림 2.1> BPMS 라이프사이클 ····· 8
<그림 2.2> Weber et al. 의 동적 프로세스의 특징 ····· 13
<그림 2.3> 웹 서비스 작동 원리 및 호출 ····· 15
<그림 2.4> SOA 환경에서 협업 프로세스 구성 예 ····· 16
<그림 2.5> 진료 프로세스의 런타임 변경 예 ····· 20
<그림 2.6> 컴포지션 토폴로지가 주어진 경우 예제 ····· 22
<그림 2.7> 직렬 통합 서비스가 존재하는 경우 ····· 24
<그림 2.8> 병렬 통합 서비스가 존재하는 경우 ····· 25
<그림 2.9> 직렬 분할 서비스가 존재하는 경우 ····· 26
<그림 2.10> 병렬 분할 서비스가 존재하는 경우 ····· 26
<그림 3.1> 기업간 협업 프로세스 ····· 30
<그림 3.2> Oh의 XPDL과 WS-BPEL의 상호 연동 예 ····· 33
<그림 3.3> 데이터 오브젝트 처리 예 ····· 34
<그림 3.4> BPMS의 웹 서비스 기반 협업 프로세스 구성 ····· 37
<그림 3.5> 제안 프레임워크를 활용한 u-헬스케어 시나리오 38
<그림 3.6> 프로세스 오브젝트 데이터 매칭 개념적 관계도 ·· 40
<그림 4.1> 웹 서비스-리소스 네트워크 ····· 45
<그림 4.2> 사이클이 존재하는 경우 ····· 51
<그림 4.3> 후방 검색 알고리즘의 개요 ····· 57
<그림 4.4> 특정 목표 리소스 생성을 위한 Greedy Set Cover 의사코드 ····· 59
<그림 4.5> 후방 검색 의사코드 ····· 60
<그림 4.6> 최소신장트리 알고리즘 적용을 위한 네트워크 수정 예 ····· 65
<그림 4.7> Edmonds 알고리즘 ····· 67
<그림 4.8> 1단계 COVER1 집합이 가능해를 보장하지 않는 경우 예 ····· 68
<그림 5.1> 네트워크 크기에 따른 수행 시간 1 ····· 75

<그림 5.2> 네트워크 크기에 따른 수행 시간 2 ····· 76
<그림 5.3> 네트워크 크기에 따른 수행 시간 3 ····· 76
<그림 5.4> 네트워크 복잡도에 따른 수행 시간 ····· 77

1. 서론

1.1. 연구 배경

BPM(Business Process Management, 비즈니스 프로세스 관리)은 초창기 IT 솔루션의 하나로 시작하여, 최근에는 기업 및 조직 전반의 업무 활동 개선을 지원하는 패러다임으로 발전해 왔다. BPM은 EAI(Enterprise Application Integration)의 연장선에서 조직 내·외부의 모든 업무 기능을 통합적으로 관리하는 체계를 제공하는 동시에 워크플로우(Workflow)와 프로세스 관리의 측면에서 조직의 지속적인 혁신을 지원한다. 특히 기업 구조가 핵심 역량 중심으로 재편되고, 아웃소싱 형태의 협업이 강조됨에 따라 기업간으로 확장된 업무 프로세스를 효과적으로 관리하는 것이 BPM의 주요 기능으로 강조되고 있다.

BPM에서 기업간 협업 프로세스 구축을 지원하는 기술로는 웹 서비스(Web Service)가 있다. 웹 서비스는 광범위 네트워크와 기업들의 어플리케이션을 통합하는 기술로서, 웹 서비스 인터페이스를 통하여 독립적으로 제작된 각 기업들의 단위 업무들을 하나의 프로세스로 결합시킨다(Lei and Duan 2007, Peltz 2003). 웹 서비스 기술을 이용하여 네트워크로 연결된 여러 기업들은 기업 내부의 업무와 외부의 웹 서비스로 대치된 업무들로 협업 프로세스(Collaborative Process)를 구성할 수 있고, 서로 다른 컴퓨팅 환경을 갖는 여러 분산 조직들은 상호운용성(Interoperability)을 보장받는다 (Shen *et al.*, 2007). 이러한 높은 상호운용성을 바탕으로 Staffware (Dumas *et al.*, 2005), WebSphere Process Server (Kloppmann *et al.* 2008), ADEPT2 (Reichert *et al.*, 1998, Reichert *et al.*, 2005), YAWL (van der Aalst and ter Hofstede, 2005)과 같은 워크플로우 시스템과 FLOWer (Dumas *et al.*, 2005, Mutschler

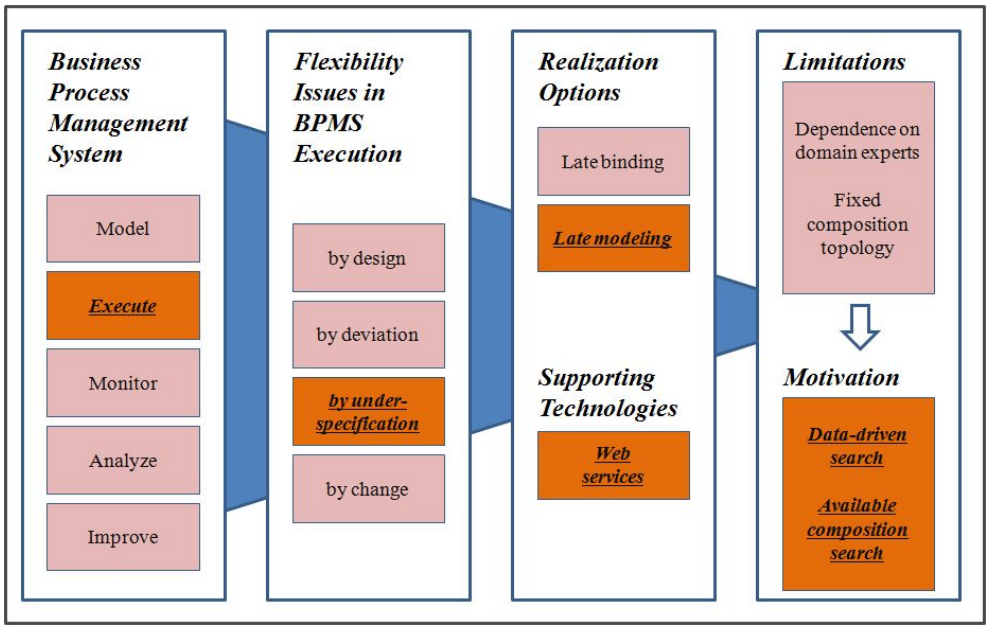
et al., 2008b, van der Aalst *et al*., 2005)과 같은 사례처리 시스템이 개발되었다.

하지만 이러한 몇몇 성공 사례가 있음에도 불구하고 BPM과 같은 프로세스 중심의 정보시스템이 널리 활용되지는 못하고 있는데, 그 가장 큰 이유 중 하나는 실시간 프로세스 변화에 대한 민첩한 지원을 못하고 있기 때문이다 (Mutschler *et al*., 2008a, van der Aalst and Jablonski, 2000). 또한 지금까지 프로세스 모델링에 사용되는 XPDL, BPEL, BPMN과 같은 표준들은 프로세스를 표현하는 측면에 초점을 맞추고 있으며, 유연성과 관련된 부분을 지원함에 미흡한 점을 보인다는 것도 BPM과 같은 프로세스 중심의 정보시스템이 널리 보급되는데 걸림돌이 되는 요소이다. 또한 이러한 유연성과 관련된 이슈는 런타임시에 발생하는 여러 예외 상황과 불확실성에 대응하기 위해서도 필요하다 (Schonenberg *et al*., 2008). 따라서 BPM 시스템에서 유연성을 높이기 위해서는 런타임시 현재 인스턴스의 내용을 기반으로 하여 프로세스 모델을 동적으로 구성할 수 있는 방안이 필요하다. 동적으로 프로세스를 구성하기 위해서는 느슨하게 기술된(Loosely Specified) 프로세스 구성 요소들을 동적으로 완성해가는 형태를 생각해 볼 수 있으며, 본 연구에서는 이를 위한 기술로 웹 서비스를 활용하여 협업 프로세스에서 실시간 동적 프로세스 구성 방안을 제시한다.

1.2. 연구 목적

연구의 목적은 <그림 1.1>과 같이 도식화할 수 있다. 본 연구는 기업간 협업 프로세스를 지원하는 시스템인 BPM 시스템을 배경으로 한다. 기업 구조가 핵심 역량 중심으로 재편되고, 아웃소싱 형태의 협업이 강조됨에 따라 기업간으로 확장된 업무 프로세스를 효과적으로 관리하는 것이 BPM의 주요 기능으로 강조되고 있으며, 업무 프로세스의 범위가 기업 외부로 확장됨에 따라 유연성을 높이는 것이 협업 프로세스의 성공을 확보하는 주요 요인으로 인식되고 있다.

본 연구는 BPM 시스템의 라이프사이클에서 프로세스의 실행과 관련된 부분을 대상으로 한다. 그리고 BPMS의 실행에서 확보해야 되는 유연성의 영역 중 프로세스 인스턴스의 정보가 불확실하거나 상황이 변경된 정보의 미달(Information Underspecification)인 상황을 고려한다. 그



<그림 1.1> 연구 목적의 도출

리고 정보의 미달 상황을 해결하는 일반적인 접근 중 늦은 모델링(Late Modeling)을 지향한다.

이전부터 BPM 시스템에서 정보의 미달인 경우에서의 유연성을 확보하기 위한 여러 가지 연구들이 제시되었다. 하지만 이들은 주로 정보의 표현 차원에서 접근한 모형화에 집중을 하고 있으며 이 문제를 실제 해결하기 위한 접근 역시 대체 프로세스를 미리 구축하여 필요시 이를 호출하는 방식을 따른다. 호출되는 이러한 프로세스는 모듈화된 빌트인 프로세스(Built-In Process)라 불린다. 이러한 빌트인 프로세스를 만드는 것은 전문적인 지식을 가진 디자이너의 역량에 의존하는 요소로 인식되고 있으며, 구현의 형태 역시 프로세스 디자이너의 디자인 편의를 지원하는 부분에 초점을 맞춰왔다.

게다가 이와 같은 접근은 런타임시 발생하는 문제를 프로세스 디자이너에게 빠르게 전달하고 런타임 내에 해결하는 과정을 포함하기 어렵다. 그 뿐만 아니라 이러한 빌트인 프로세스를 활용하여 프로세스 런타임에서 발생하는 여러 상황들을 처리하기 위해서는 발생가능한 모든 상황을 미리 예측할 수 있어야 한다. 하지만 프로세스 디자이너가 이러한 상황들을 미리 예측하는 것은 현실적으로 불가능하며, 예측되지 못한 상황에 대응하기 위해서는 프로세스 인스턴스의 폐기 또는 재시작 이외의 절차를 마련하는 것이 어렵다. 따라서 빌트인 프로세스만으로 런타임시 발생하는 여러 상황을 대처하려는 접근은 불충분하며, 런타임 과정에서 인스턴스의 폐기 또는 재시작 뿐만 아니라 변환 및 우회의 가능성을 찾고 이에 해당하는 업무를 할당하는 늦은 모델링에 대한 연구가 필요하다.

이를 가능케 하는 기술적 기반으로 웹 서비스(Web Services)를 활용한다. 웹 서비스 기술 중심 SOA의 프레임워크를 통하여 상이한 컴퓨팅 환경의 기업들은 상호운용성을 보장하는 업무 자동화와 협업 프로세스 구축이 가능해졌다. 하지만 웹 서비스를 통한 협업 프로세스의 구성과

관련된 기존의 연구들은 고정된 토폴로지의 추상적인 협업 프로세스 스키마가 주어지면 이를 할당하는 방식을 따른다. 프로세스의 단순 실행의 측면에서는 고정된 토폴로지에서의 운영이 합당하다고 볼 수 있지만 협업 프로세스를 이루는 파트너의 상황이 유동적인 웹 서비스 환경에서는 고정된 토폴로지를 유지하는 것이 프로세스 유연성에 제약을 가할 수 있으며, 더 나은 파트너 서비스를 찾을 수 있는 기회를 제한할 수도 있다.

따라서 본 연구에서는 가능한 컴포지션 토폴로지를 웹 서비스 네트워크를 통하여 탐색하여 찾고, 이에 맞춰 필요시 유동적으로 협업 프로세스를 동적으로 구성하는 방법과 이를 위한 프레임워크를 제안한다.

1.3. 연구 구성

이어지는 본론부터의 논문 구성은 다음과 같다.

2장에서는 본 연구의 관련 분야인 BPMS와 업무 프로세스에서의 유연성, 그리고 프로세스 동적 구성에 활용할 웹 서비스와 웹 서비스 컴포지션에 대한 선행 연구를 살핀다. 그리고 이들 선행 연구로부터 도출한 연구 동기를 설명한다.

3장에서는 런타임 프로세스의 동적 구성을 위한 프레임워크를 제안한다. 세부적으로 웹 서비스 기반 협업 프로세스 모델과 웹 서비스를 통해 런타임 프로세스를 동적으로 구성하는 과정을 설명한다.

4장은 런타임 프로세스의 동적 구성을 위해 선택할 웹 서비스들의 최적 선택 문제를 모형화하고 이를 푸는 휴리스틱 알고리즘을 제안한다.

5장에서는 제안한 휴리스틱의 성능을 웹 서비스 네트워크의 크기와 복잡도에 따라 살펴본다.

끝으로 6장에서는 본 논문에서 다룬 연구 내용을 요약 정리하고, 그 의의와 추후 연구 방향을 논의한다.

2. 관련 연구

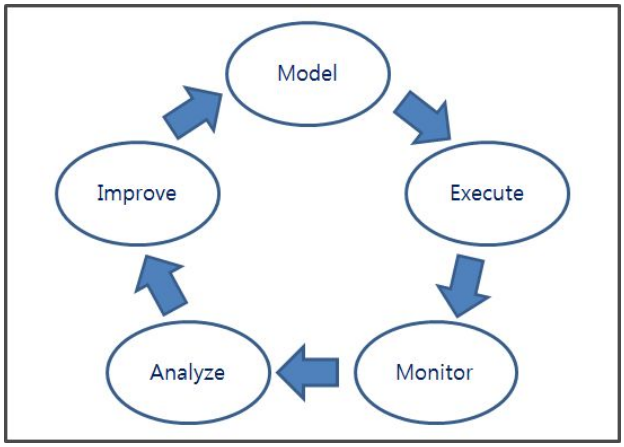
2장에서는 본 연구와 관련된 이전의 연구들을 정리한다.

우선, 연구의 배경인 BPMS(Business Process Management System)에 대해 설명한다. 그리고 BPMS에서의 프로세스 유연성과 관련된 기존 연구들을 살핀다. 이후 BPMS에서 협업 프로세스를 구축하기 위해 필요한 기술인 웹 서비스(Web Service)와 웹 서비스로 프로세스를 구축하는 웹 서비스 컴포지션에 관한 이전 연구를 기술한다. 이후 이들 연구에서 다루지 못한 부분들을 연구 동기와 관련된 절에서 제시하여, 본 논문에서 다루고자 하는 문제를 구체화한다.

2.1. Business Process Management System

BPM은 EAI(Enterprise Application Integration)의 연장선에서 조직 내·외부의 모든 업무 기능을 통합적으로 관리하는 체계를 제공하는 동시에 워크플로우(Workflow) 관리의 측면에서 조직의 반복적인 업무를 설계, 자동화하는 기술이다. 이러한 기술적 역량을 바탕으로 기업 내외부의 업무를 통합하고 관리에 사용되는 시스템을 BPMS(Business Process Management System)라고 한다 (Smith and Fingar, 2003). 이러한 BPMS는 업무 흐름의 관리와 업무 성과 평가의 가시화를 가능케 하고 업무 개선 가능성의 발굴에도 기여할 수 있어, 현장에서 높은 관심을 받았으며 이로 인하여 관련 연구도 활발히 진행되어 왔다 (Gulledge, 2007).

BPMS는 업무 전반을 모델링하는 작업에서부터 시작하여 업무의 운영 및 실행, 모니터링, 진단 및 최적화까지 일종의 라이프사이클(Life-cycle)을 이루며 지속적인 개선에 기여한다 (Weske, 2007). 이러한 BPMS의 라이프사이클에 대해 세부적으로는 연구 관점에 따라 차이를 보이지만, 일반적으로 요구되는 기능들은 다음 <그림 2.1>과 같다.



<그림 2.1> BPMS 라이프사이클

<그림 2.1>에서 정의한 BPMS 라이프사이클 각각의 단계는 다음의 기능을 수행한다.

- *Model* : 업무 프로세스를 정의하고 이를 모형화하는 단계로서, 업무를 수행하는데 필요한 모든 요소들을 반영한다. 작성된 프로세스 모형은 BPMS 내의 저장소에서 관리된다.

- *Execute* : 모형화된 프로세스가 BPMS 엔진을 통하여 실행되는 단계를 의미한다. BPMS는 프로세스 내 특정 업무와 관련을 가지는 어플리케이션의 작동을 제어한다.

- *Monitor* : BPMS 엔진에 의해 수행된 업무의 결과를 추적하는 단계로서, 모형화 단계에서 정의된 주요성과지표(Key Performance Indicator) 및 업무 구성 요소들의 값을 측정한다.

- *Analyze* : 모니터링 단계에서 수집된 평가지표 및 구성 요소들의 결과값을 분석하는 단계이다. 프로세스에서 발생한 병목지점이나 낮은 효율을 보이는 곳을 찾고, 그 원인을 분석한다.

- *Improve* : 분석 단계에서 발견한 문제를 해결하고, 최적화 알고리즘 및 시뮬레이션 등을 이용하여 개선 방안을 구체화하는 단계이다. 이를 통하여 프로세스의 지속적인 개선에 기여한다.

이러한 기능들로 인하여 BPMS는 RTE(Real-Time Enterprise)로의 전환을 위한 중요한 솔루션으로 인식되고 있으며 (Sinur and Bell, 2003), 이를 통하여 기업은 변화의 강도와 속도가 증가하고 있는 경영환경에서 자신의 경쟁력을 유지・강화하고자 한다 (Lam *et al.*, 2009).

또한 최근 기업 구조가 핵심 역량 중심으로 재편되고, 아웃소싱 형태의 협업이 강조됨에 따라 기업의 업무는 조직 외부로 확장되고 있다. 이처럼 기업의 업무 범위가 확장됨에 따라 조직간 협업의 효율성이 중요해

지고, 확장된 업무 프로세스를 효과적으로 관리, 지원하는 것이 BPMS의 주요 기능으로 강조되고 있다. 기업마다 상이한(Heterogeneous) 컴퓨팅 환경에서 조직간 협업 프로세스를 구축하기 위해서는 시스템의 상호운용성을 보장해야 하는데, 이를 위한 프레임워크로 서비스 지향 구조(SOA, Service Oriented Architecture)가 있다. SOA 프레임워크에서 각 협업을 수행할 조직의 업무는 서비스로 구현이 되며, 해당 서비스에 대한 상세한 기술을 담은 API를 제공하는 방식이 아닌, 통신을 위한 표준화된 프로토콜의 인터페이스만으로 연결되는 느슨한 결합(Loosely Coupled)을 지향한다 (Singh and Huhns, 2005).

하지만 이러한 서비스 지향 구조의 느슨한 형태의 업무 결합은 업무 범위의 확장가능성을 높인다는 긍정적인 측면은 있지만, 런타임시에 발생하는 여러 예외 상황과 불확실성에 대한 위험은 더 커졌다고 볼 수 있다. 기업 내부에서 프로세스 전반을 관장하는 경우에는 여러 상황에 대한 모니터링을 직접 수행하여 실시간 대응이 가능하지만, 서비스 지향 구조에서는 정의된 메시지에서 규정한 내용 이외의 요소는 직접 확인이 불가능하기 때문이다. 따라서 외부 조직과의 협업 프로세스에서 발생하는 불확실성에 대응할 수 있는 유연성 확보가 더욱 중요해졌음을 알 수 있다 (Schonenberg *et al.*, 2008).

2.2. 런타임 프로세스의 유연성

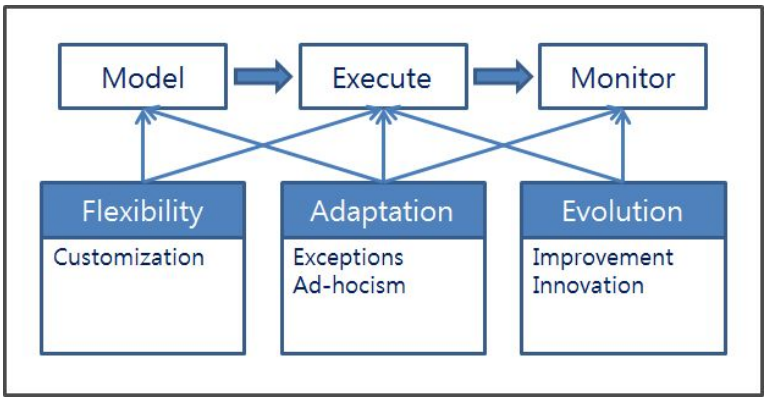
워크플로우와 프로세스 기술 커뮤니티에서는 업무 환경 변화에 대한 적응을 위한 요소로 프로세스의 유연성(Flexibility)을 오랫동안 중요하게 인식해 왔다 (Heinl *et al*., 1999, Reijers 2006). 유연성에 대한 정의는 어떤 조직의 능력과 변화에 대응하는 지원 기술에 따라 다르게 표현될 수 있다. Regev와 Wegmann은 프로세스의 유연성을 파악하기 위하여 반대의 관점에서 무엇이 변하지 않고 그대로 남아있는지에 초점을 맞추었다 (Regev and Wegmann, 2005). 또한 어떤 프로세스가 다른 새로운 프로세스로 완전히 대체되지 않은 상태에서 일부 바뀔 수 있으면 유연하다고 판단할 수 있다는 주장하였으며 (Regev *et al*., 2006), 프로세스의 유연성은 환경 변화와 안정 사이에서 주어진 업무를 변함없이 수행하는 효율적 균형 유지라고도 정의하였다 (Regev *et al*., 2007).

모든 환경 변화가 시스템에 영향을 미치는 것은 아니므로 van der Aalst와 Jablonski (van der Aalst and Jablonski, 2000)는 워크플로우 시스템이 유연성을 확보할 필요가 있는 변화의 유형을 파악하고자 하였다. 그 결과 유연성을 요구하는 변화에 대한 여섯 가지 판단 기준을 제시하였고, 이를 정리하면 다음 표와 같다.

[표 2.1] van der Aalst and Jablonski의 프로세스 변화 판단 기준

기준	내용
변화의 원인	시장의 변화 법적 규제 기술 환경의 변화 내부 시스템의 논리적 디자인 오류 시스템 기술 문제
변화의 영향	일시적으로 영향을 미치는 변화 시스템 전반의 개선을 야기하는 변화
변화의 대상	프로세스 조직 구조 자료 구조와 정보 업무 기능 업무 통합
변화의 종류	업무, 역할의 확장 업무, 기능의 축소 업무, 기능의 대체 업무, 기능의 재설계 및 조정
변화의 허용시점	진입시에만 허용 항상 허용
프로세스 인스턴스의 대응 방안	폐기 및 다음 인스턴스의 진행 롤백 또는 종료 후 재시작 강제 진행 인스턴스의 변환 인스턴스의 우회

이러한 환경 변화에 능동적으로 대처할 수 있는 프로세스를 동적 프로세스(Dynamic Process)라고 부르며, 동적 프로세스를 구축하기 위해서는 유연성, 적응력, 진화가능성을 갖추어야 한다 (Weber *et al.*, 2009). 동적 프로세스의 요구사항과 프로세스 라이프사이클 내의 각 단계의 관계는 다음 <그림 2.2>와 같다.



<그림 2.2> Weber *et al.* 의 동적 프로세스의 특징

- *유연성(Flexibility)* : 프로세스 내의 모든 단계와 상황을 자세히 모델링하여 완성된 형태의 프로세스로 고정시키지 않고, 파편화된 부분 프로세스를 구축하여 느슨한 형태의 프로세스를 런타임 과정에서 호출하여, 완성한다.

- *적응력(Adaptation)* : 다양한 예외 상황에서도 프로세스 인스턴스가 처리될 수 있도록 이를 기술적으로 지원할 수 있어야 한다.

- *진화(Evolution)* : 다양한 변화의 상황에서 프로세스 인스턴스를 처리한 과정 및 결과를 모니터링하여 프로세스의 개선에 활용될 수 있어야 한다.

동적 프로세스를 얻기 위하여 런타임 프로세스 유연성이 갖춰야 할 체계에 대해서는 이전부터 많은 연구가 있어 왔다. Snowdon (Snowdon *et al.*, 2007)은 유연성 확보가 필요한 세 가지 인자로 정보 유형에 따른 유연성, 양에 따른 유연성, 구조적 유연성을 꼽았다. Soffer (Soffer, 2005)는 프로세스의 유연성을 표준 절차에 의한 업무 수행에 일시적으로 영향을 미치는 단기 유연성과 업무 전반에 영향과 관련을 맺는 장기 유연성으로 구분하였다. Kumar와 Narasipuram (Kumar and Narasipuram, 2006)은 디자이너에 의해 작성된 미리 정의된 유연성(predefined

flexibility)과 런타임에서 발생하는 여러 변형을 처리하기 위해 필요한 요소를 구분하였다. Carlsen *et al.* (Carlsen *et al.*, 1997)은 워크플로우 시스템에서 요구되는 유연성을 업무 품질 평가 프레임워크를 사용하여 살펴보았으며, Heinl *et al.* (Heinl *et al.*, 1999)은 두 가지 분류 스키마로 선택에 의한 유연성과 적응에 의한 유연성을 제안하였다. Heinl *et al.* 이 제안한 선택에 의한 유연성은 프로세스 모형화 단계에서 디자이너에 의해 정의된 프로세스를 인스턴스의 상황에 맞춰 선택하는 방식을 따르고, 적응에 의한 유연성은 프로세스 상에 발생한 새로운 요구 사항에 맞춰 워크플로우를 새로이 적응시키는 과정을 요구한다. 선택에 의한 유연성을 확보하기 위해서는 업무 분석을 토대로 빌트인(Built-In) 프로세스를 구축하고 상황에 맞춰 적절한 빌트인 프로세스를 호출하여 인스턴스의 실행 경로를 결정해야 한다. 이러한 빌트인 프로세스를 구축하는 것은 실행 이전에 결정되는 사전 모형화(Advance Modelling) 방법이다. 반면 적응에 의한 유연성은 실행 도중에 발생하는 상황에 맞춰 프로세스에 대응책을 제시하는 늦은 모형화(Late Modelling)에 해당한다.

Schonenberg *et al.* (Schonenber *et al.*, 2008) 은 이들 연구를 토대로 런타임 프로세스의 유연성을 크게 네 가지로 분류하였다.

- *디자인(design)에 의한 유연성* : 디자인에 의한 유연성은 런타임 프로세스의 인스턴스를 변경해야 할 때, 변경 작업을 최소화하기 위한 설계 유연성을 확보하는 것을 의미한다. 이를 위해 여러 개의 프로세스 스키마를 미리 디자인하여 해당 인스턴스의 상황에 맞는 프로세스를 제공하는 것을 목표로 한다.

- 편차(deviation)에 대한 유연성 : 편차에 대한 유연성은 프로세스 내 단위 업무에 할당된 인스턴스 처리 순서에 대한 유연성을 의미한다. 주어진 프로세스에서 특정 단위 업무를 프로세스에서 정의된 순서와 상관없이 먼저 처리할 수 있도록 변경을 허용하는 것을 의미한다. 특정 단

위 업무에서 처리해야 할 인스턴스의 수가 많아질 경우, 병목을 막기 위하여 이와 같은 순서 변경에 대한 지원이 필요하다.

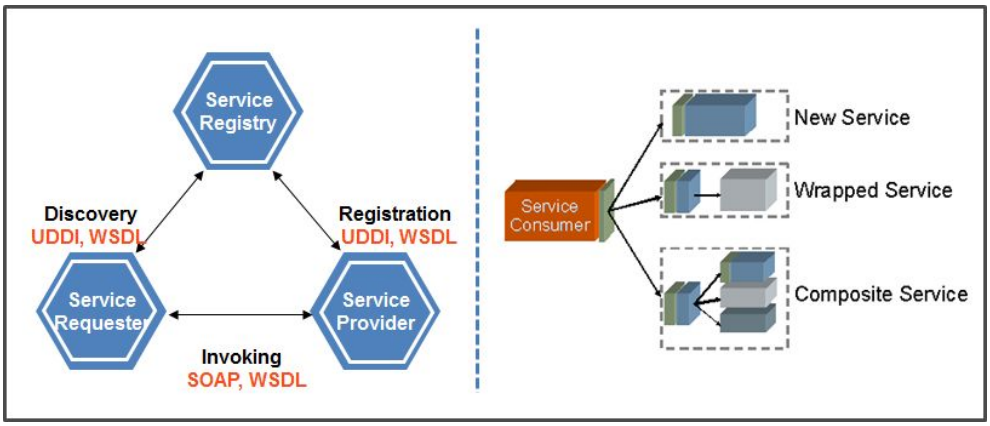
- 정보 미달(underspecification)에 대한 유연성 : 정보 미달에 대한 유연성은 불완전한 프로세스를 완성해 나갈 수 있는 유연성을 의미한다. 현재 특정 인스턴스를 처리할 수 있는 프로세스가 존재하지 않을 때 이를 대처하기 위하여 프로세스를 능동적으로 변화시킬 수 있는 능력을 의미한다.

- 변경(change)에 의한 유연성 : 현재 인스턴스가 따르고 있는 프로세스 스키마를 새로운 프로세스로 변경할 수 있는 유연성을 의미한다. 이는 다른 유연성과 달리 프로세스 스키마의 명세 자체를 변화할 수 있는 적응성이 필요하며, 여러 개의 프로세스 버전이 필요한 경우에 유연성을 높일 수 있는 접근이다.

본 연구에서 제안하는 방법은 런타임 프로세스 인스턴스의 정보 미달인 경우에 유연성을 높일 수 있는 방안을 다룬다.

2.3. 웹 서비스와 웹 서비스 컴포지션

웹 서비스(Web Service)는 웹 상에서 정의된 모듈화된 소프트웨어 컴포넌트로서, 개방형 표준 데이터 표현 기법인 XML(eXtensible Markup Language)과 표준화된 인터넷 프로토콜을 결합시킨 분산 컴퓨팅 기술이다 (Alonso *et al.*, 2004). 웹 서비스는 W3C, OASIS와 같은 국제 표준화 기구들이 제창한 표준을 바탕으로 높은 상호운용성(Interoperability)을 보장한다. 웹 서비스의 작동 원리 및 호출의 모습은 다음 <그림 2.3>과 같다.

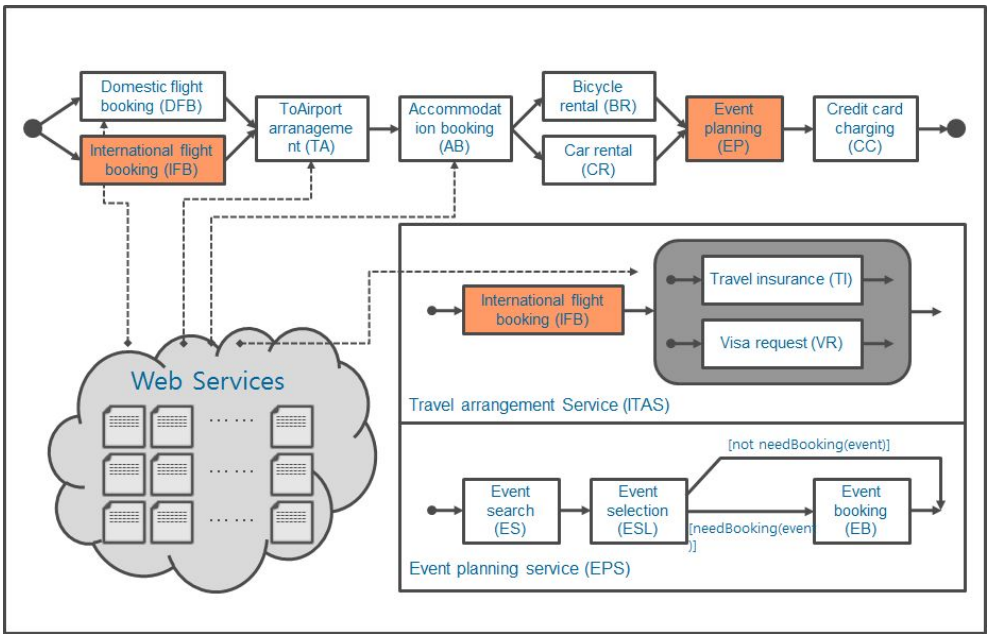


<그림 2.3> 웹 서비스 작동 원리 및 호출

웹 서비스의 구동을 위하여 <그림 2.3>에 명시된 표준 WSDL(Web Service Description Language), UDDI(Universal Description Discovery and Integration), SOAP(Simple Object Access Protocol)을 활용한다. WSDL은 서비스 공급자가 자신이 공급하는 서비스의 내용을 기술할 수 있도록 한다 (Christensen *et al.*, 2001). UDDI는 서비스 공급자는 WSDL로 기술한 웹 서비스를 등록하고, 서비스 사용자가 이를 검색할 수 있도록 하는 표준화된 방법을 제공한다 (Bellwood *et al.*, 2003). 그리고 SOAP은 서비스 사용자가 웹 서비스를 선택하고, 실제 웹 서비스 공

급자와 통신을 할 때 따라야 할 메시지의 형태 및 구성 방식 등에 대한 표준화된 프로토콜이다 (Box *et al.*, 2000).

또한 웹 서비스들을 호출하기 위하여 만들어진 원격 프로시져를 서비스화 함으로써 서비스 지향 구조(SOA, Service Oriented Architecture)를 구현할 수 있다. 상호운용성을 보장하는 웹 서비스를 활용하는 SOA의 프레임워크를 통하여 기업은 내외부의 파트너들과의 업무 자동화 및 협업 프로세스 구축이 용이해진다. 다음 <그림 2.4>는 SOA 환경에서 외부의 웹 서비스들을 활용하여 협업 프로세스[1)]를 구성한 모습을 보여준다.



<그림 2.4> SOA 환경에서 협업 프로세스 구성 예

이처럼 상호운용성을 보장하는 웹 서비스에 의해, 상이한 시스템의 프로세스 구축 가능성이 확보된다. 그리고 웹 서비스로 이루어진 프로세스

1) 웹 서비스들의 연쇄로 이루어진 서비스를 컴포지트 웹 서비스(Composite Web Service)라 부르기도 하고, 이는 협업 프로세스와 대응된다.

를 더욱 효과적으로 관리하기 위해 여러 연구와 표준들이 고안되었다. WfMC(Workflow Management Coalition)에서 제안한 프로세스 교환 표준인 XPDL(XML Process Definition Language)은 단위 업무로 웹 서비스를 포함할 수 있도록 지원하고, BMI DTF (Business Modeling & Integration Domain Task Force)에서 제창한 프로세스 모델링 기법 표준인 BPMN(Business Process Modeling Notation)은 웹 서비스 실행 언어 표준인 WS-BPEL(Web Service Business Process Execution Language)로의 변환을 지원한다. 또한 Chi *et al.* (Chi et al., 2008)는 이러한 표준들을 활용한 Petri Net 기반의 SCMP(Service Composition Modelling Platform)를 제시하고, SCMP의 시각화, 문서변환, 워크플로우 측정 모듈을 제안하여 웹 서비스를 활용한 프로세스 시스템 구축을 지원하였다.

협업 프로세스를 구축하기 위해서는 외부의 웹 서비스를 직접 활용해야 되므로 협업 프로세스의 품질은 웹 서비스의 QoS 관리와 직접 연결된다. 협업 프로세스 구성을 위해 호출된 웹 서비스의 QoS와 관련된 연구들은 단위 웹 서비스들의 QoS가 어떤 식으로 프로세스 구조와 맞물려 결합해야 하는지를 다룬다. Cardoso *et al.* (Cardoso *et al.*, 2004)는 협업 프로세스의 QoS를 계산하기 위하여 대표적인 제어 흐름인 직렬, 병렬, 조건, 순환에 대한 QoS 지표별 계산 방법론을 제시하였고, Jaeger *et al.* (Jaeger *et al.* 2004)는 워크플로우 패턴을 이용한 QoS 계산 방법을 제시하였다.

협업 프로세스에서 단위 업무들의 품질 결합과 관련된 부분을 웹 서비스 QoS에서 다루었다면, 협업 프로세스 내 단위 업무를 실제 담당할 웹 서비스를 결정해야 하는 문제를 해결해야 한다. 이를 웹 서비스 선택 문제(Web Service Selection Problem)라 한다. QoS 속성들 사이에는 일종의 트레이드오프(Tradeoff) 관계를 갖는 경우가 존재한다. 그 예로는 저렴한 실행 비용인 웹 서비스의 경우 결과물의 품질 수준이 떨어질 수

있고, 처리 시간이 빠른 웹 서비스는 실행 비용이 높은 경우를 생각할 수 있다. 웹 서비스 선택 문제는 여러 QoS 속성을 가지는 웹 서비스들이 존재하는 환경에서 사용자의 요구와 트레이드오프 관계를 고려하여 최적의 웹 서비스를 찾는 문제이다.

여러 연구들에서 웹 서비스 선택을 위한 최적화 문제를 모형화하였다 (Yu *et al.*, 2007, Yuan *et al.*, 2009, Zeng *et al.*, 2004). 이 최적화 문제들은 다항시간안에 종료되지 않는 NP-Hard 형태로 나타나는데, 해당 문제들에 적절한 유사최적해를 제공하기 위한 별도의 휴리스틱을 제시하거나, 유전 알고리즘(Genetic Algorithm) 등의 메타 휴리스틱을 활용해 그 결과를 적용하는 연구들이 있다 (Canfora *et al.*, 2005).

일반적인 웹 서비스 QoS의 최적화 문제 이외에도 협업 프로세스에 대응되는 컴포지트 웹 서비스의 QoS를 전역적(Global) 관점과 지역적(Local) 관점에서 동시에 평가하거나 (Ardagna and Pernici, 2005), 웹 서비스 선택 기준에 사용자 선호도를 적용하는 연구 (Sun *et al.*, 2007), 퍼지이론을 적용한 연구 (Wang *et al.*, 2006) 등 다양한 현실적 요구를 반영한 문제들이 정의되고, 이를 푸는 알고리즘이 제시되었다. 또한 컴포지트 웹 서비스의 실행을 보장하기 위하여 Ardagna와 Mirandola (Ardagna and Mirandola, 2010)는 미리 정의된 여러 BPEL 프로세스 각각에 여러 번의 요청을 함으로써 프로세스의 최적화를 실시간으로 하는 방법을 제시하였고, Oh *et al.* (Oh *et al.*, 2011)는 컴포지트 웹 서비스의 실행을 보장 전략을 위하여 시간적, 물리적 중복을 적용한 전략을 제시하였고, 이를 풀기 위한 유전 알고리즘 기반 휴리스틱을 제안하였다.

2.4. 연구 동기

2.4.1. 런타임 프로세스 운영 관점

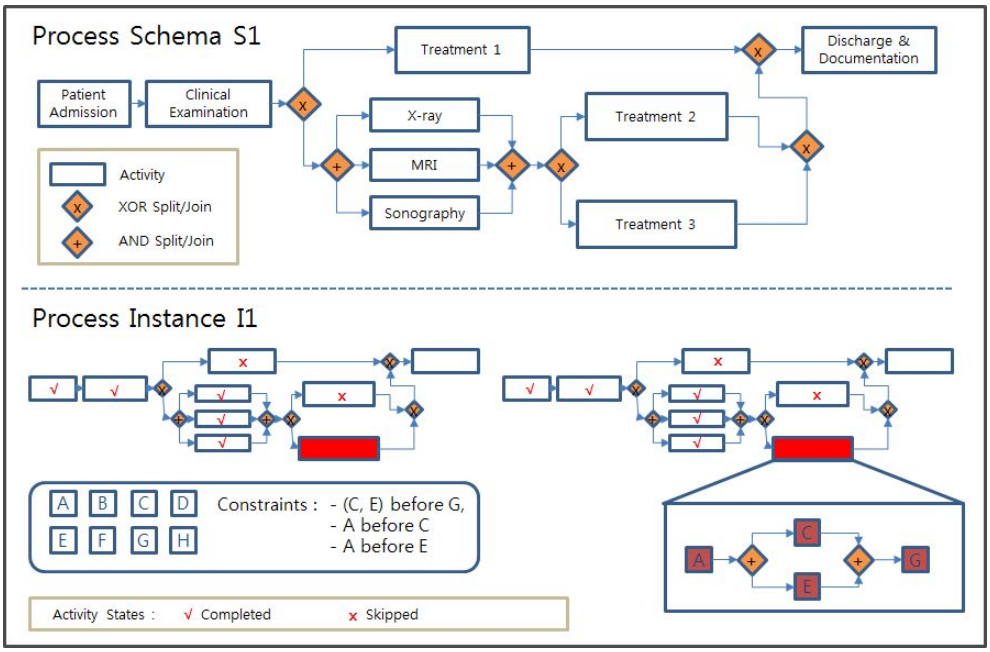
앞에서 프로세스의 유연성을 필요로 하는 변화의 상황과 이를 해결하기 위한 유연성의 체계 정립에 관련된 기존의 연구들이 소개되었다. 여기에는 프로세스의 변화를 표현할 수 있는 스키마 개발과 이를 활용하는 메카니즘에 대한 연구가 포함된다.

이러한 업무 변화의 상황에서 변화할 업무의 내용을 어떻게 결정할 것인가에 대한 연구는 미흡한 실정이다. 대부분의 연구들은 미리 구축된 모듈화된 빌트인 프로세스를 호출하는 방식을 취하는데 이러한 빌트인 프로세스를 만드는 것은 전문적인 지식을 가진 디자이너에 의해 결정되는 요소로 인식되고 있고, 이를 지원하기 위한 기술적인 부분에만 초점을 맞춰왔다 (R. Aguilar-Saven, 2004). 일부 Min *et al.* (Min *et al.*, 2010)과 같이 의사결정지원 시스템의 관점에서 현재 업무의 상태 정보에 근거한 단위 업무 결정 지원 방법이 제시되기도 하였다. 하지만 이는 모듈화된 내부 프로세스의 경우에만 지원 가능한 형태이며, 본 연구에서 다루고 있는 협업 프로세스를 구성하기 위해서는 여러 대안 중에서 하나를 선택할 수 있는 QoS 관련 기준이 적용되어야 한다.

이러한 빌트인 프로세스를 활용하여 프로세스 런타임에서 발생하는 여러 상황들을 처리하기 위해서는 발생가능한 모든 상황을 미리 예측할 수 있어야 한다. 하지만 프로세스 디자이너가 이러한 상황들을 미리 예측하는 것은 현실적으로 불가능하다. 이러한 경우 예측되지 못한 상황에 대해서는 프로세스 인스턴스의 폐기 또는 재시작 이외의 절차를 마련하는 것이 힘들다. 따라서 빌트인 프로세스만으로 런타임시 발생하는 여러

상황을 대처하는 것은 불충분하며, 런타임 과정에서 인스턴스의 폐기 또는 재시작 뿐만 아니라 변환 및 우회의 가능성을 찾고 이에 해당하는 업무를 할당하는 늦은 모델링에 대한 연구가 필요하다.

또한 프로세스 인스턴스가 런타임 도중에 현재의 프로세스에 적합하지 않다는 것을 확인한 경우, 해당 인스턴스의 처리를 위해 프로세스를 수정하는 작업을 거쳐야 한다. 다음 그림과 같이 진료 프로세스에서 다른 치료 절차로 변경해야 할 상황이 생겼을 때를 생각해보자.



<그림 2.5> 진료 프로세스의 런타임 변경 예

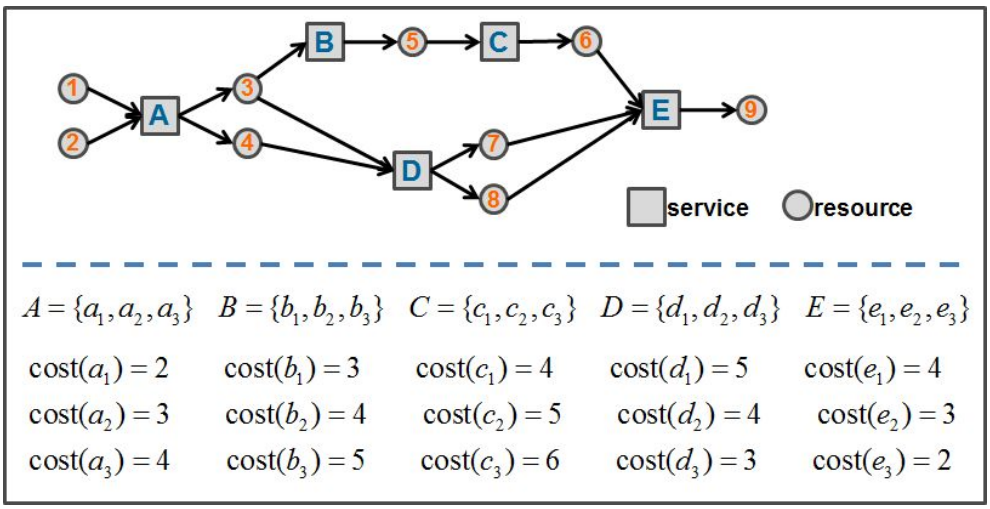
<그림 2.5>는 기존의 Treatment 3에 해당하는 업무를 실행하여 진료를 계획하였으나, 인스턴스의 속성이 이에 적합지 않다고 판단을 한 경우이다. 이 때, 대체되는 진료 업무 G를 수행해야 하는데, G와 관련된 빌트인 프로세스는 없는 상태이다. 이러한 상황에서 업무 G를 수행하기 위해서는 제약에 의해 업무 C와 E를 먼저 실행해야 하고, 업무 C, E 각

각은 직전 수행 업무로 업무 A를 할당해야 한다. 그러므로 현재 인스턴스의 상황에 의해 기존 Treatment 3을 변경하기 위해서는 진료 업무 G를 수행하기 이전에 업무 A, C, E의 선후행 제약 관계를 고려한 서브 프로세스를 위와 같이 작성할 수 있는 기반이 마련되어야 한다.

따라서 본 연구에서는 이와 같이 업무를 변경해야 할 필요가 생겼을 때, 이를 대체할 업무를 웹 서비스를 통해 구조화하여 선택할 수 있는 기반을 제공하고자 한다. 이를 위해 호출할 웹 서비스들의 기술된 파라미터 정보를 활용하여 웹 서비스들 사이의 데이터 오브젝트 중심의 인과관계를 그래프로 표현한다. 표현된 그래프에서 현재 인스턴스와 대응하는 데이터 오브젝트를 출발점으로 삼아 목표한 단위 업무 웹 서비스 혹은 필요한 웹 서비스 오퍼레이션 파라미터를 획득할 수 있는 가능성을 탐색한다.

2.4.2. 웹 서비스 컴포지션 측면

기존의 협업 프로세스를 위한 웹 서비스 컴포지션과 관련된 연구들은 대부분 프로세스의 형태, 즉 컴포지션 토폴로지(Composition Topology)가 고정된 경우를 가정한다. 고정된 컴포지션 토폴로지를 사용하여 웹 서비스를 선택하는 것은 빌트인 프로세스를 미리 정의하고 이에 해당하는 단위 웹 서비스를 호출하는 방식이다. 하지만 런타임 프로세스 운영 관점에서 설명하였듯이, 모든 상황에 맞는 빌트인 프로세스를 미리 만드는 것은 현실적으로 불가능하며, 설령 고정된 토폴로지를 가진 빌트인 프로세스가 존재하더라도 이는 더 나은 웹 서비스를 선택할 수 있는 기회를 제한할 수 있다. 여기서는 몇 가지 경우를 통해 특정 컴포지션 가능성을 놓치는 경우를 설명하고자 한다.



<그림 2.6> 컴포지션 토폴로지가 주어진 경우 예제

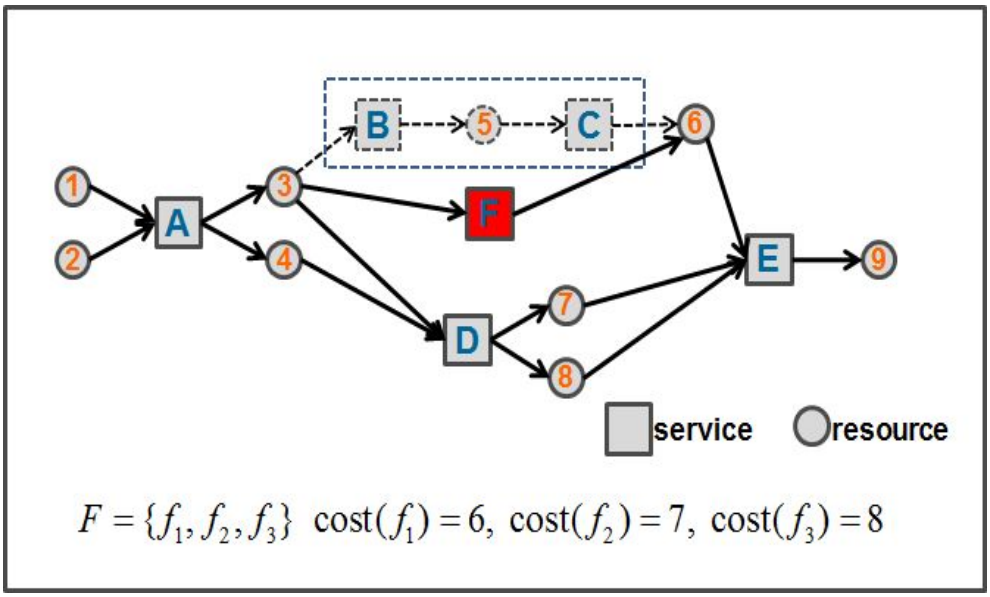
웹 서비스를 기반으로 한 <그림 2.6>과 같은 토폴로지의 추상적인 프로세스가 주어져 있다고 생각해보자. <그림 2.6>에서 사각형으로 그려진 노드는 프로세스 내의 단위 업무를 지칭하고, 원으로 그려진 노드는 해

당 업무의 수행 전후에 연관된 데이터 오브젝트, 조건 등을 의미한다. 여기서는 웹 서비스로 이루어진 업무들을 대상으로 하므로, 원으로 표현되는 이들 노드들은 웹 서비스가 수행하는 역할과 관련된 메시지 타입(Message Type), 선후행 조건, 웹 리소스 등과 매칭되는데, 이들 리소스는 WSDL에 기술된 내용을 추출하여 얻을 수 있는 정보들이다 (이후부터는 웹 서비스와 실행 전후 상황과 관련을 맺는 요소들을 리소스라 칭한다). 예를 들어, <그림 2.6>의 업무 A에 할당되는 웹 서비스는 리소스 1, 2를 사용하여, 리소스 3, 4를 만들어 내는 업무를 수행한다.

위와 같은 리소스와 업무의 관계로 이루어진 추상적인 협업 프로세스가 주어져 있을 때, 협업 프로세스를 만든다고 생각해보자. 그러면 주어진 토폴로지로 표현되는 협업 프로세스의 단위 업무 A, B, C, D, E의 역할을 담당하는 실제 웹 서비스를 선택해야 한다. <그림 2.6>의 상황에서 선형 비용 결합이 가능한 웹 서비스들이라고 가정하였을 때, 최소비용의 협업 프로세스를 얻으려면, 업무 A에 할당될 수 있는 웹 서비스 a_1, a_2, a_3 중 가장 적은 비용을 사용하는 a_1을 호출하고, 같은 방식으로 업무 B에 b_1, 업무 C에 c_1, 업무 D에 d_3, E에 e_3를 호출하는 방식을 취할 수 있다. 따라서 협업 프로세스는 5개의 웹 서비스 a_1, b_1, c_1, d_3, e_3로 이루어진다. 하지만 이와 같이 고정된 토폴로지 하에서 웹 서비스를 할당하는 방식은 다음과 같은 웹 서비스가 있는 경우에 최적의 협업 프로세스를 구축할 수 없다.

(1) 통합 서비스가 존재하는 경우 (Merging Case)

통합 서비스가 존재하는 경우는 기존의 컴포지션 토폴로지에서 단위 업무들을 포함하는 새로운 서비스가 있는 경우다.

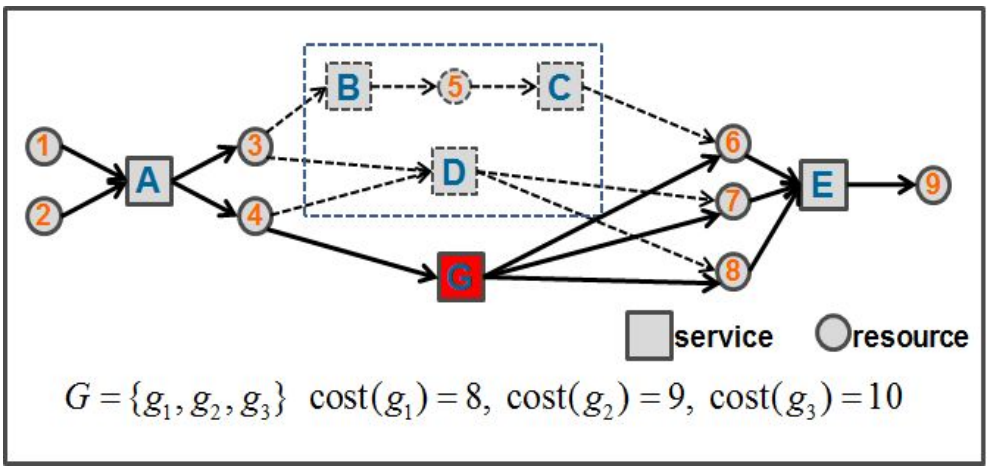


<그림 2.7> 직렬 통합 서비스가 존재하는 경우

만약 리소스 3을 입력 요소로 사용하고, 리소스 6을 생성하는 어떤 웹 서비스들이 존재한다고 가정해보자. 그러면 이 웹 서비스들은 <그림 2.7>의 협업 프로세스 상에서 F의 위치에 놓일 수 있다. 그리고 F의 위치에 놓일 웹 서비스의 비용이 업무 B와 C 각각에 호출된 웹 서비스들의 비용합보다 저렴할 경우, 직렬로 연결된 두 개의 서비스를 호출하는 것보다 F에 해당하는 하나의 서비스를 호출하는 것이 더 효율적이다. <그림 2.6>에서 B와 C에 할당될 수 있는 최소 비용의 웹 서비스는 각각 b_1, c_1이고, 이들의 비용합은 7이다. <그림 2.7>에서 보인 비용이 6인 웹 서비스 f_1을 호출하게 되면, 이전의 직렬 구조로 연결된 업무 B와 C를 고정하여 실행했을 때보다 1단위 저렴한 협업 프로세스를 구축할 수

있다.[2)]

병렬 형태의 통합 서비스가 존재할 경우에도 마찬가지이다.



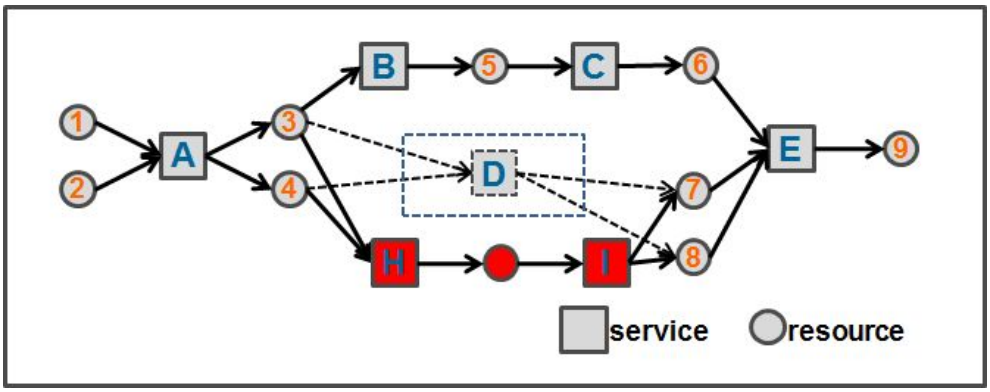
<그림 2.8> 병렬 통합 서비스가 존재하는 경우

<그림 2.8>은 업무 B, C, D와 입력 요소는 달리하지만, 업무 B, C, D의 병렬 업무의 최종 결과는 같은 새로운 업무 G와 같은 역할을 수행하는 웹 서비스들이 존재하는 경우이다. 업무 G는 리소스 4를 사용하여 리소스 6, 7, 8를 만들고, 이 중 서비스 g_1은 B, C, D 에 할당된 웹 서비스 b_1, c_1, d_3의 비용합 10보다 비용 저렴하다. 따라서 <그림 2.6>의 업무 B, C, D에서 병렬적으로 진행될 세 개의 서비스를 호출하지 않고, 단일 서비스 g1만을 호출하여 협업 프로세스를 구성하는 것이 더 나은 프로세스 구성 방향이 된다.

2) 서비스 f_1은 UDDI 상에서는 하나의 웹 서비스로 등록·배포되나, 실제 내부 실행은 여러 웹 서비스들로 구성되어 있는 컴포지트 웹 서비스 형태로 포장(wrapped)될 수 있다.

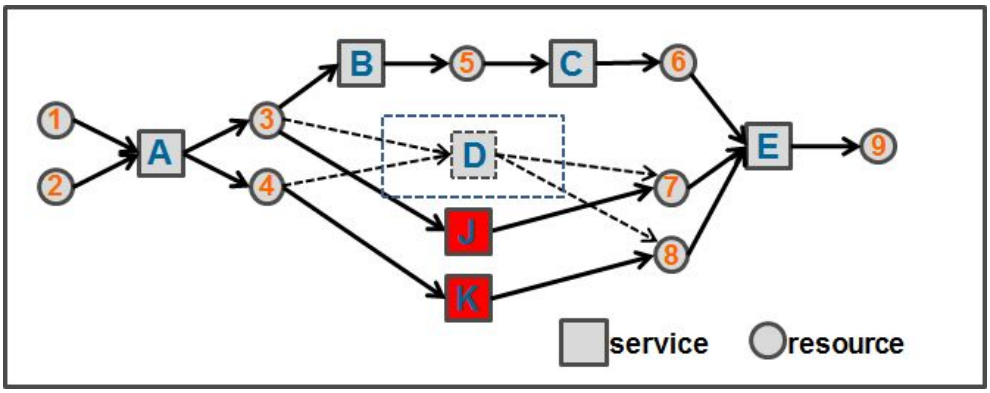
(2) 분할 서비스가 존재하는 경우 (Splitting Case)

분할 서비스가 존재하는 경우는 하나의 단위 업무에 하나의 웹 서비스를 호출하여 처리되던 일을 직렬 혹은 병렬 형태로 업무를 나누어서 웹 서비스를 분할 호출할 수 있는 경우다. <그림 2.9>은 직렬 분할로 하나의 웹 서비스에서 처리하던 업무 D를 두 개의 웹 서비스 H와 I를 순차적으로 실행시켜 처리하는 것을 보여준다.



<그림 2.9> 직렬 분할 서비스가 존재하는 경우

마찬가지로 <그림 2.10>은 하나의 웹 서비스를 통해 수행되던 업무 D를 두 개의 웹 서비스 J와 K를 병렬로 실행시켜 처리하는 모습을 보여준다.



<그림 2.10> 병렬 분할 서비스가 존재하는 경우

(3) 리소스 사용에 제약이 발생하는 경우

앞에서 리소스는 웹 서비스가 수행하는 역할과 관련된 메시지 타입, 선후행 조건, 웹 리소스 등을 포괄적으로 표현한 것이라고 정의하였다. 이 리소스는 협업 프로세스의 단위 업무를 수행하는 과정에서 발생하는 중간 결과물임과 동시에 다음의 업무를 수행하기 위해 필요한 입력 요소로 활용된다. 리소스 사용에 제약이 발생하는 경우는 이 리소스에 문제가 발생한 경우이다. 고정된 컴포지션 토폴로지의 협업 프로세스의 실행 과정에서 단위 업무가 정상적으로 종료되지 않거나, 생성된 중간 결과물이 요구한 품질 수준에 미치지 못할 경우가 발생할 수 있다. 이런 경우는 일종의 예외 상황으로 간주할 수 있는데, 해당 프로세스 인스턴스는 별도의 조치를 취할 때까지 대기하거나 직전 작업 또는 전체 프로세스를 재시작하는 과정을 일반적으로 거치게 된다. 하지만 이러한 처리는 협업 프로세스의 수행 시간을 길게 만드는 단점이 있으며, 현실적으로도 발생할 수 있는 모든 예외 상황을 사전에 모두 고려하는 것은 불가능하다. 따라서 원래 계획된 협업 프로세스를 실행하다가도, 현재 인스턴스의 리소스 상황이 여의치 않을 시에도 실행을 보장할 수 있는 협업 프로세스를 구축할 수 있는 방안이 필요하다.

(4) 빌트인 프로세스 구축을 위한 지침으로 활용

프로세스 동적 구성 방법에 대한 연구는 웹 서비스 기반의 협업 프로세스의 디자인 과정을 지원하는 정형화된 방법을 제공할 수 있다. 프로세스 디자인과 관련하여 분석 기법들은 BPR(Business Process Reengineering) 등의 연구를 통해 많이 다루어져 왔지만, 실제 협업 프로세스의 디자인 자체를 지원하는 수리화된 방법은 드물다 (Bradley *et al.*, 1995). 그 이유는 일반적인 기업 프로세스의 경우 디자인의 차원에서 고려해야 할 요소들 중 정형화하기 어려운 것들이 존재하여 대안이 될 프로세스 디자인 자체를 평가하는 과정에서 개념적 혼란을 야기할 수 있

기 때문이다.

하지만 웹 서비스로 이루어진 협업 프로세스의 경우에는 수행 비용, 결과물의 품질 등 정량적이면서 명확한 QoS 속성들만 고려할 수 있다. 이는 일반적인 기업 내부 프로세스의 경우에는 업무 흐름에서 다른 기능과의 연계가능성, 문화적인 요소와 같은 의미론적인 측면이 부각될 수 있지만, 웹 서비스 중심의 협업 프로세스의 경우에는 의미론적으로는 느슨한 결합(Loosely Coupled)을 지향하기 때문이다. 따라서 프로세스 디자이너는 웹 서비스들을 선택하여 프로세스 대안의 가능성을 탐색하고, 이를 통해 기존의 협업 프로세스를 개선할 수 있는 기회를 획득할 수 있다. 즉 프로세스 스키마로부터 웹 서비스 컴포지션을 일방적으로 강요당하는 것이 아닌, 컴포지션 가능성에 대한 사전 탐색 기회를 얻음으로써 프로세스 디자이너는 빌트인(Built-In) 프로세스 스키마 구축에 활용할 자료를 얻을 수 있다.

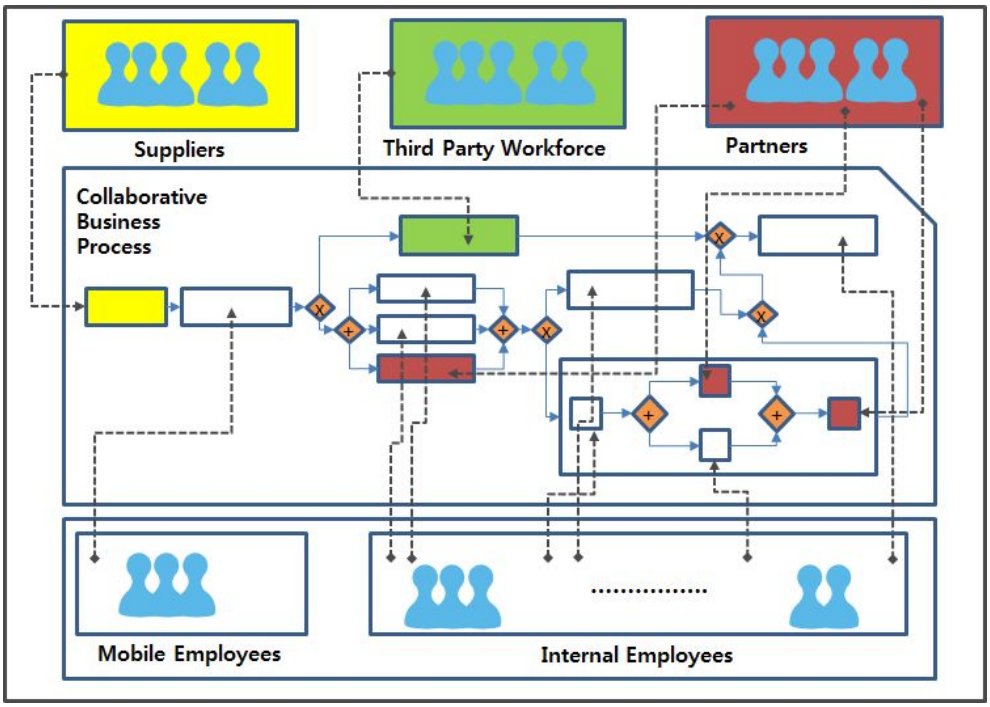
앞에서 언급한 (1), (2), (3) (4)의 경우들과 같이 고정된 컴포지션 토폴로지로 협업 프로세스를 구축하는 것은 웹 서비스 환경에서 얻을 수 있는 더 좋은 품질의 업무 실행 가능성 및 프로세스 디자인 개선 가능성을 놓칠 수 있다. 따라서 본 연구에서는 현재 프로세스 인스턴스 결과를 바탕으로 사용가능한 리소스를 정형화하고, 웹 서비스들의 네트워크를 사전 탐색하여 컴포지션 토폴로지와 단위 업무로 기능할 웹 서비스의 선택을 동시에 고려하는 늦은 모델링 방법을 제안한다.

3. 런타임 프로세스 동적 구성 프레임워크

본 장에서는 웹 서비스 기반의 협업 프로세스를 동적으로 구성하는 프레임워크를 제안한다. 우선 협업 프로세스를 구성할 수 있는 기반인 웹 서비스 기반 협업 프로세스 모델을 제시한다. 그리고 제시된 협업 프로세스 모델에서 기업 내부 프로세스와 외부 웹 프로세스 사이의 상호운용성 보장을 다룬다. 이후 런타임시 프로세스 스키마를 변경해야 할 때 웹 서비스를 활용하여 해결하는 과정과 이에 필요한 늦은 모델링 모듈(Late Modeling Module)에 대해 설명한다.

3.1. 웹 서비스 기반 협업 프로세스 모델

협업 프로세스는 기업 내부의 어플리케이션들 간의 통신으로 이루어지는 기업 어플리케이션 프로세스(Enterprise Application Process)에서 출발하였다. 기업 어플리케이션 프로세스는 기업내 정보의 양이 방대해져 어플리케이션 단위의 관리가 어려워지는 상황에서, 업무 이벤트에 대한 효용성과 재사용성을 높이기 위하여 만들어졌다. 기업 어플리케이션 프로세스를 구축하기 위해서는 다수의 어플리케이션 및 그 데이터를 통합해야 하며, 데이터 형식과 코드의 변환을 담당하는 데이터 브로커와 메시징 입출력을 담당하는 어댑터가 구현되어야 한다. 이런 기업 어플리케이션 프로세스의 범위를 기업 외부로 확장한 것이 협업 프로세스(Collaborative Business Process)이다.



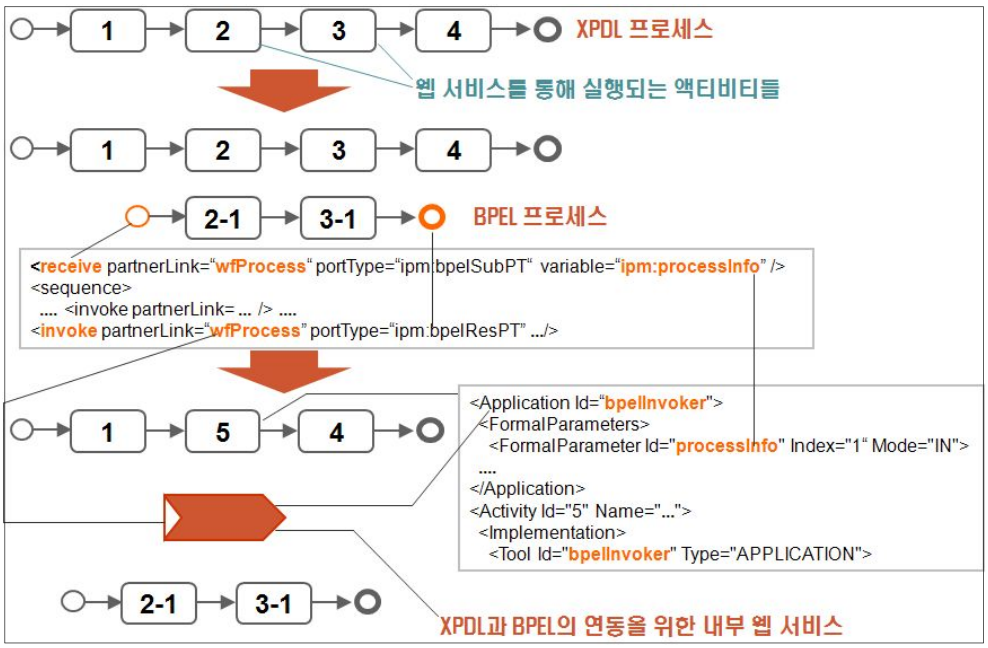
<그림 3.1> 기업간 협업 프로세스

상이한 컴퓨팅 환경에서 운영되는 여러 기업들을 협업 프로세스에 참여시키기 위해서는 이들을 연결할 수 있는 범용의 인터페이스가 필요하다. 웹 서비스 기술은 각 기업의 다양한 응용 프로그램들에 동일한 인터페이스를 제공하고, 기업간 협업 프로세스의 설계를 용이하게 만들어 파트너와의 협업을 이룰 수 있는 근간이 된다. 웹 서비스 기술을 이용하여 <그림 3.1>과 같이 공급자 및 파트너, 타사의 외부 용역들과도 협업 프로세스를 구성할 수 있으며, 모바일을 비롯한 원격 기기를 활용하는 업무 환경에서도 내부 직원들이 참여를 지원한다.

이처럼 웹 서비스를 통하여 모든 협업을 위한 메시지의 교환이 이루어진다 해도, 협업 파트너들이 제공하는 웹 서비스들의 입출력 파라미터의 타입, 품질 수준은 다르다. 따라서 동일한 목적과 구조를 지닌 프로세스라 하더라도 파트너에 따라 상이한 조정 과정을 거쳐야만 한다. 즉, 동일한 목적과 구조를 가진 프로세스라 하더라도 파트너가 달라짐에 따라 기업은 협업 프로세스의 내용을 새롭게 구성해야 해야 한다. 이러한 제약 하에서는 협업을 위한 파트너 선정에 제약이 따르고, 협업 프로세스 과정이 복잡해진다. 따라서 웹 서비스 기반의 협업 프로세스에서 기업 내부의 워크플로우 프로세스와 웹 서비스로 이루어진 프로세스를 연결하기 위해서는 파트너와 독립적으로 운영될 수 있는 환경이 보장되어야 한다. 또한 런타임 과정에서 웹 서비스 프로세스를 호출하기 위한 조건이 불충분하거나, 해당 웹 서비스의 입출력 파라미터가 변경되어 정상적인 수행이 불가능한 상황을 표현하기 위해 인스턴스의 정보와 웹 서비스의 입출력 파라미터를 연결하는 방법 역시 필요하다.

3.1.1. 워크플로우 프로세스와 웹 서비스 프로세스의 파트너 독립성 보장

본 절에서는 워크플로우 프로세스와 웹 서비스 프로세스의 파트너 독립성을 보장하는 방법을 설명한다. 워크플로우 프로세스와 웹 서비스 프로세스들이 혼합된 전체 프로세스를 프로세스 정의 언어 표준인 XPDL(XML Process Definition Language)로 우선 정의하고, 이 중 웹 서비스 프로세스는 효율적 실행을 위해 XPDL로 정의된 임의 업무와 연결되는 프로세스로 관리하며, 해당 프로세스를 WS-BPEL(Web Service Business Process Execution Language)로 표현한다. XPDL과 WS-BPEL은 각각 WfMC(Workflow Management Coalition), OASIS에서 제안된 표준들이며, 이들 표준은 서로 다른 단체들에서 제안되었기 때문에 서로의 상호운용성에 대해서는 각각의 문서들에서 다루어지지 않고 있다. 다만 별도의 스키마로 정의된 데이터 오브젝트를 호출하고, 이를 할당할 수 있는 틀을 마련함으로써 적절한 변환 및 연결 과정을 거쳐 상호운용 가능성을 열어두고 있다. Oh (Oh, 2011)는 XPDL과 WS-BPEL의 상호 연동 시나리오를 제시하여 각각의 표준으로 기술된 협업 프로세스의 상호운용의 가능성을 연구하였다. 이 시나리오는 다음 <그림 3.2>와 같다.

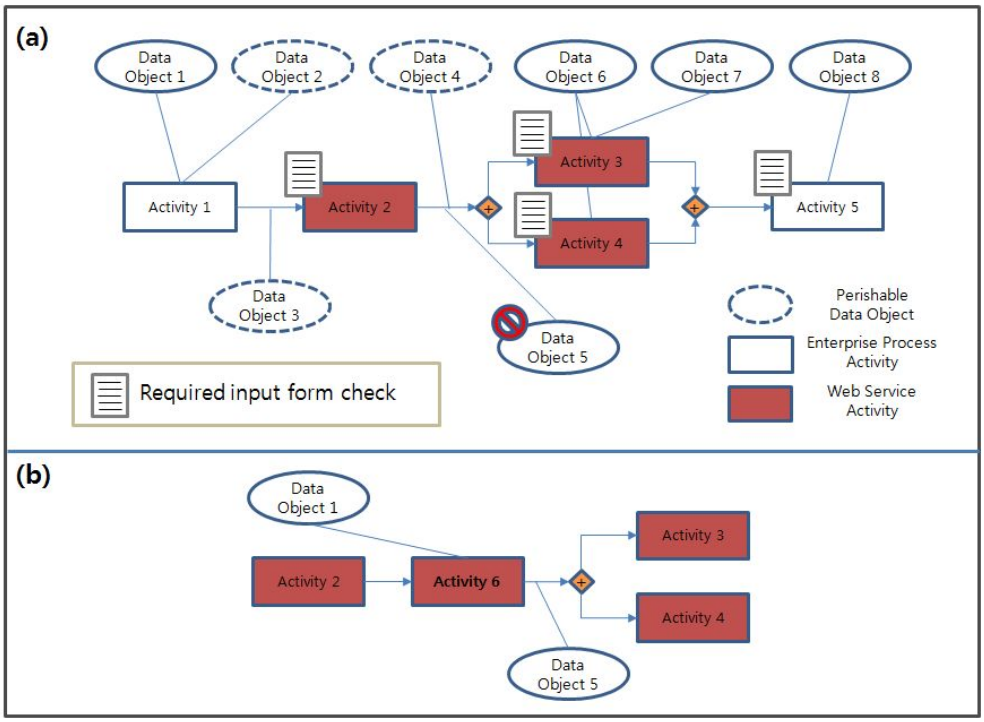


<그림 3.2> Oh의 XPDL과 WS-BPEL의 상호 연동 예

<그림 3.2> 예의 주어진 XPDL 프로세스에서 업무 2와 3은 외부 파트너에 의해 실행될 서브 프로세스로 결정된 상황이다. 그러면 업무 2와 3으로 이루어진 서브 프로세스를 WS-BPEL로 매핑함으로써, 웹 서비스 구성을 완성한다. <그림 3.2>의 연동의 예에서 업무 2와 3을 연결한 서브 프로세스에 해당하는 WS-BPEL 프로세스가 작성되고, XPDL 프로세스에서는 업무 2와 3을 wfProcess 라는 파트너로부터 수행됨을 명시한다. 다음 작성된 WS-BPEL 프로세스와 XPDL 프로세스를 연결하기 위해 bpelInvoker라는 어플리케이션을 활용한다. 따라서 실제 1-2-3-4로 정의된 프로세스는 워크플로우 프로세스의 관점에서는 업무 2와 3을 포함하는 업무 5로 대치된다. 그리고 업무 5는 별도의 WS-BPEL 프로세스 실행을 명령하고, WS-BPEL 프로세스가 종료된 후에 작업을 마치게 된다. 이와 같은 시나리오를 통해, 전체 프로세스 내에서 웹 서비스 실행 환경을 독립적으로 이루어질 수 있도록 한다.

3.1.2. 인스턴스 정보와 웹 서비스 입출력 파라미터의 연결 보장

런타임 과정에서 웹 서비스 프로세스를 호출하기 위한 조건이 불충분하거나, 해당 웹 서비스의 입출력 파라미터가 변경되어 정상적인 수행이 불가능한 상황을 표현하기 위해 인스턴스의 정보와 웹 서비스의 입출력 파라미터를 연결하는 방법이 필요하다. 이를 위해서 프로세스 내의 각 업무의 시작 전에 데이터 오브젝트의 유효성을 검사한다. 데이터 오브젝트 처리 과정의 예는 다음 <그림 3.3>과 같다.



<그림 3.3> 데이터 오브젝트 처리 예

<그림 3.3(a)>에서 업무로 향하는 호와 연결된 데이터 오브젝트는 업무의 입력 요소로 사용되는 것을 지칭하며, 업무와 업무 사이를 잇는 호

와 연결된 데이터 오브젝트는 중간 생성물에 해당한다. 이를 따르면 업무 1은 데이터 오브젝트 1과 2를 입력으로 사용하여 데이터 오브젝트 3을 생성하고, 업무 2에 해당하는 웹 서비스를 호출하여 프로세스를 진행한다. 업무 2는 웹 서비스를 호출하기 전에 먼저 입력으로 들어온 데이터 오브젝트의 유효성을 검사한다. 업무 2의 경우, 호출할 웹 서비스가 결정이 되면 업무 1의 수행 결과인 데이터 오브젝트 3의 내용이 해당 웹 서비스의 입력 파라미터로 사용될 수 있는지 여부를 확인한다. 이를 위해 WSDL에 기술된 웹 서비스의 입출력 파라미터를 사용하여 구현된 WS-BPEL 프로세스의 입출력 데이터 표현법과 XPDL의 데이터 표현법과의 상호운용성이 보장되어야 한다. 상호운용성에 대한 보장은 앞 절의 내용을 따른다.

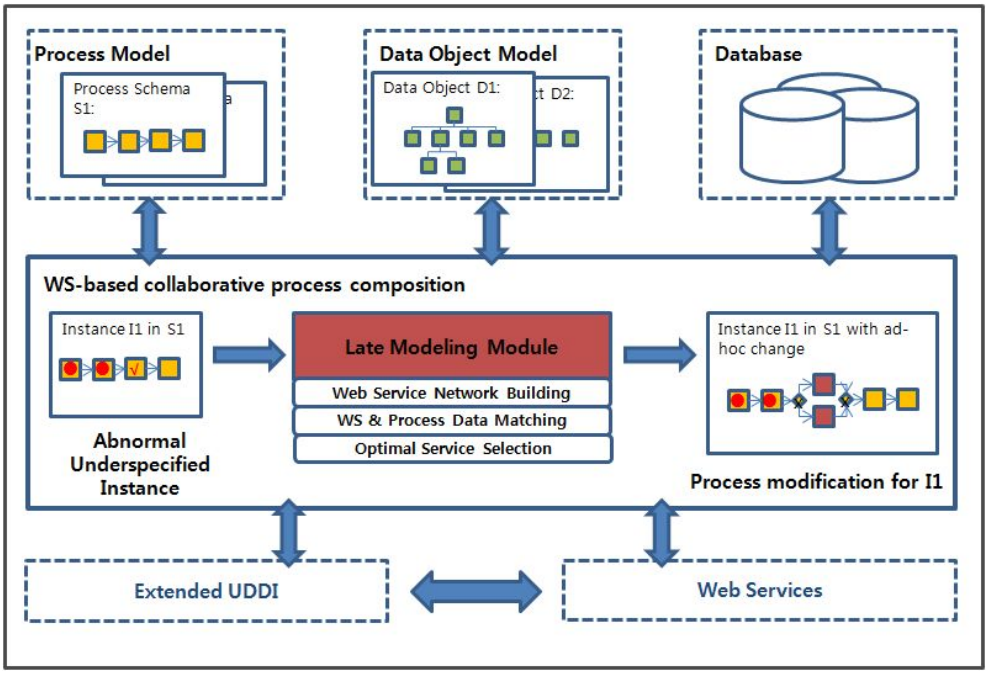
<그림 3.3(a)>의 프로세스를 계속 진행한 결과를 살펴보면, 다음 업무 3과 4를 실행하고자 하는데, 유효성 검사에서 업무 3과 4에서 필요한 데이터 오브젝트 4, 5 중 5를 사용할 수 없는 경우가 발생하였다. 그러면 이전 업무 2를 다시 재시작하는 방법을 생각할 수 있지만 주어진 예에서 업무 2의 입력인 데이터 오브젝트 3은 재사용이 불가능한 소멸되는 데이터이다. 그러면 다시 데이터 오브젝트 3을 생성하기 위하여 업무 1의 재시작을 고려할 수 있지만 업무 1의 입력인 데이터 오브젝트 2도 마찬가지로 재사용이 불가능한 소멸되는 데이터여서 업무 2의 시작과 동시에 가용성을 잃었다. 그러면 이 프로세스 인스턴스는 종료를 하고, 데이터 오브젝트 2를 얻기 위한 별도의 프로세스를 모색해야 한다. 이와 같은 처리는 납기시기를 맞추는 것이 중요한 프로세스인 경우 높은 비효율을 낳는다.

다음 절에서 설명할 웹 서비스를 이용한 프로세스 동적 구성은 <그림 3.3(b)>에 제시한 가능성을 탐색한다. 현재 업무 2를 수행한 결과 중 유효성 검사를 통과한 데이터 오브젝트 4는 다음 업무 3, 4에 사용될 수 있는 상태이다. 업무 3, 4를 수행하기 위해서는 유효한 데이터 오브젝트

5가 필요하므로 현재 인스턴스에서 사용 가능한 데이터 오브젝트인 1을 활용하여 해당 오브젝트를 만드는 웹 서비스를 UDDI 검색을 통해 찾는다. 그리고 만약 이러한 웹 서비스가 존재한다면 기존 프로세스에서 동적으로 업무 6이 새로 삽입이 되고 이를 업무 3, 4와 AND JOIN으로 연결한다. 그리고 새로 추가된 업무 6은 늦은 모델링 모듈이 생성한 WS-BPEL 프로세스로 수행됨을 명시하여 상호운용을 보장한다.

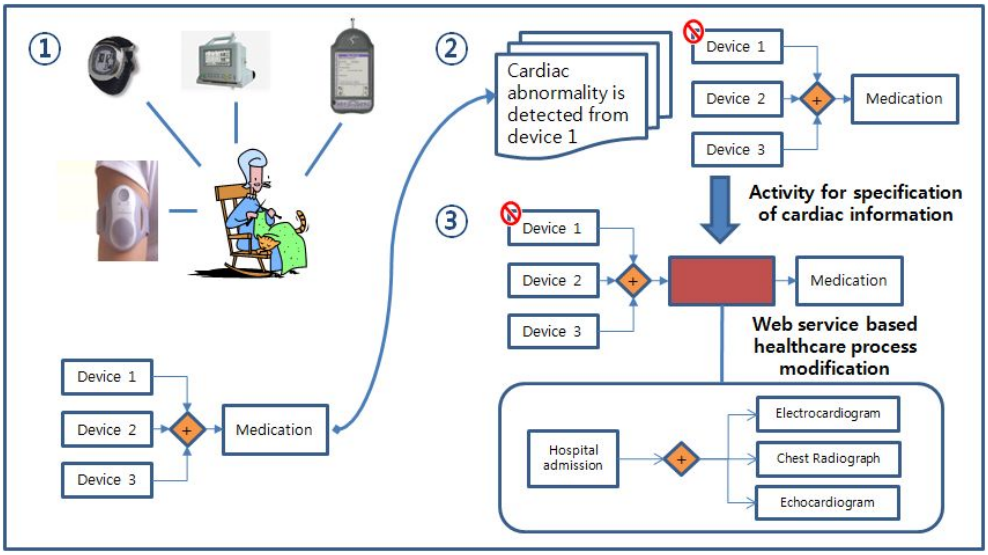
3.2. 웹 서비스를 이용한 런타임 프로세스 구성

웹 서비스 기반의 협업 프로세스를 구성하기 위하여 <그림 3.4>와 같이 BPMS 실행 엔진에 웹 서비스 컴포지션을 지원하는 늦은 모델링 모듈(Late Modeling Module)을 작성한다.



<그림 3.4> 웹 서비스 기반 협업 프로세스 구성 프레임워크

제시한 웹 서비스를 이용한 협업 프로세스 구성 프레임워크를 활용한 예를 위해 다음 <그림 3.5>와 같은 가상의 u-헬스케어 서비스 시나리오를 고려할 수 있다.



<그림 3.5> 제안 프레임워크를 활용한 u-헬스케어 시나리오

<그림 3.5>의 시나리오는 유비쿼터스 환경에서 헬스케어 서비스를 이용하는 고령의 환자 케이스를 다루고 있다. 이 환자는 의료 정보를 모니터링할 수 있는 스마트기기를 사용하고 있으며, 이들 기기들을 이용한 상태 측정값들을 토대로 약물의 종류와 복용량을 결정하는 헬스케어 프로세스를 따르고 있다. 그런데 환자의 심장과 관련된 정보를 모니터링하는 device 1에서 약물과 관련된 결과를 도출하는데 사용할 수 없는 비정상(abnormal) 값이 나온 경우, 이 환자에게 적절한 약물 처방(Medication)을 내리는 다음 작업을 수행할 수 없게 된다. 따라서 이러한 경우 적절한 약물 처방을 결정하는데 미달된 정보인 심장과 관련된 양상을 확인하는 절차가 필요하다. 이를 위해 헬스케어 서비스는 약물 처방 이전 단계로 가상의 웹 서비스 프로세스를 작성한다. 이 가상의 웹 서비스 프로세스는 환자의 심장에 대한 의학적 검사를 수행할 병원을 검색 및 선택하고, 약물 처방에 필요한 파라미터를 제공하는 검사만을 선택적으로 구성한다.

위 시나리오와 같이 웹 서비스 기반의 협업 프로세스 구성 프레임워크는 프로세스 인스턴스에 발생한 정보 미달의 경우에 대처하기 위해 별도의 프로세스를 생성하고 실행하는 것이 아니라, 현재의 목적을 달성하기 위한 서브 프로세스를 구성하여 실행 중인 프로세스를 완성하는 것을 지향한다.

웹 서비스 기반의 협업 프로세스를 동적으로 구성하기 위하여 늦은 모델링 모듈(Late Modeling Module)은 웹 서비스 네트워크 구축, 인스턴스 정보 매핑, 최적 웹 서비스 선택 작업을 수행한다. 이들에 대한 자세한 설명은 다음 절에서 소개한다.

3.2.1. 웹 서비스 네트워크 구축

협업 프로세스를 구성할 웹 서비스를 선택하기 위해서는 어떠한 웹 서비스가 존재하는지를 파악할 수 있어야 한다. 이를 위하여 확장 UDDI를 활용하여 웹 서비스 저장소에 등록된 WSDL 정보를 획득하고, 이 정보를 바탕으로 세부 웹 서비스들의 명세를 확인한다. 그리고 웹 서비스들 사이의 관계를 네트워크로 구성한다.

WSDL로 기술된 웹 서비스는 오퍼레이션들과 각 오퍼레이션의 입력 파라미터, 출력 파라미터로 이루어져 있다고 정의할 수 있다. 그러면 다음과 같은 정의가 가능하다. 어떤 웹 서비스에 오퍼레이션 op가 있고, op는 입력 파라미터의 집합 $IN=\{i_1, i_2, \ldots, i_m\}$과 출력 파라미터 $OUT=\{o_1, o_2, \ldots, o_n\}$을 갖는다.

또한 웹 서비스의 오퍼레이션은 일반적으로 네 가지 유형 (1) one-way: 호출 요청은 받지만 답은 하지 않음, (2) request-response: 호출 요청을 받고, 답을 함, (3) solicit-response: 요청을 보내고, 답이 오기를 기다림, (4) notification: 메시지를 보내기만 함 (Chinici *et al.*, 2007)이 존재하는데, 본 연구에서는 협업 프로세스 구성에 사용될 수 있는 (2)와 (3)만을 대상으로 고려한다.

이러한 웹 서비스들로 네트워크를 구축하기 위해서는 어떤 웹 서비스의 입출력 요소가 다른 웹 서비스의 입출력 요소들과 어떤 관계를 맺고 있는지 파악하는 작업이 필요하다. 서로 다른 공급자가 자신만의 방식으로 웹 서비스의 파라미터와 오퍼레이션을 기술하고, 이들 각각은 독자적인 입출력 요소와 유형을 갖을 수 있기 때문에 서로 다르게 기술된 표현의 의미적 연결을 파악하는 것이 중요하다. 이를 해결하기 위한 방법으로 자연언어처리(Natural Language Processing)를 활용한 어근추출 방

법과 폭넓은 의미를 반영하기 위한 RDF(W3C, 2004), WSDL-S(Akkiraju *et al.*, 2005)를 활용할 수 있다. 본 연구에서는 기술된 표현들 사이의 의미적 연결을 위한 유사도 측정 기술은 제공되어 있다는 가정하에 웹 서비스 내 오퍼레이션의 파라미터 비교를 통한 매칭을 사용하였다. 이를 정형화하면 다음과 같다.

· 정의 1. (파라미터 유형) A boolean function, pType(p_1.type, p_2.type) returns True if: (1) p_1.type=p_2.type, or (2) p_1.type can be derived from p_2.type in WSDL or RDF or WSDL-S.

· 정의 2. (파라미터 이름) A boolean function, pName(p_1.name, p_2.name) returns True if: S(p_1.name, p_2.name)$\geqq\Theta$, where S a is similarity measuring function, and Θ is a given threshold.

· 정의 3. (파라미터 매칭) A boolean function pMatch(p_1, p2, S, Θ) returns True if: (1) pType(p_1.type, p_2.type)=True, and (2) pName(p_1.name, p_2.name)=True.

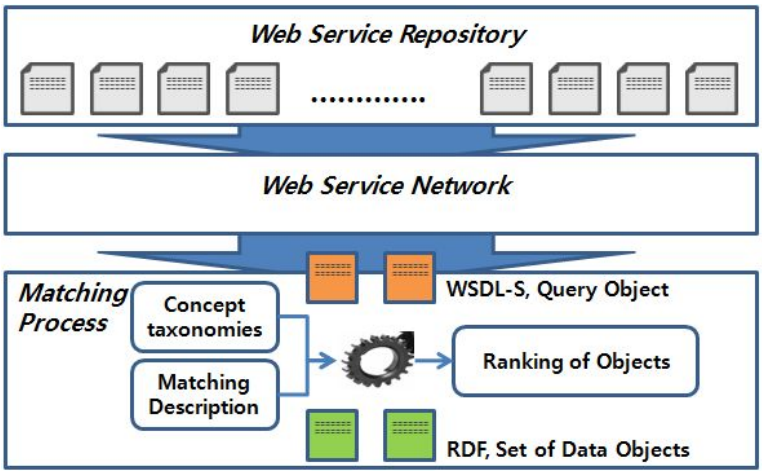
파라미터들간의 의미를 매칭하는 과정 이후에는 실제 웹 서비스의 기능을 담당하는 오퍼레이션들이 파라미터를 매개로 어떠한 형태로 연결되었는지 파악해야 한다. 오퍼레이션 호출 관계는 다음과 같이 정의된다.

· 정의 4. (오퍼레이션 호출) For two operations $op_1(IN_1, OUT_1)$ and $op_2(IN_2, OUT_2)$, (1) op_1 full invocation op_2, denoted by "$op_1 \rightarrow op_2$" if $i_2 \in OUT_1$, $\forall i_2 \in IN_2$, (2) op_1 partial invocation op_2, denoted by "$op_1 \rightharpoonup op_2$" if $i_2 \in OUT_1$, $\exists i_2 \in IN_2$

3.2.2. 프로세스 데이터 오브젝트 매칭

앞 절에서는 웹 서비스 저장소에 등록된 웹 서비스들 사이의 관계를 네트워크로 표현하는 과정을 설명하였다. 이를 통하여 웹 서비스 네트워크를 구축하였으면, 현재 인스턴스 데이터 오브젝트가 가진 정보의 내용이 네트워크 상의 어떤 입출력 파라미터와 관련있는지 비교, 확인하는 작업이 필요하다.

이를 위해 프로세스에서 발생할 수 있는 데이터 오브젝트의 내용과 제약 사항들을 표현할 수 있는 체계를 구축한다. 그리고 이들 데이터 오브젝트와 웹 서비스의 파라미터들 사이의 의미상 유사도(Semantic Similarity)를 측정한다. 예를 들어 웹 서비스의 파라미터에 대한 내용은 WSDL-S 하에서 구현되고 데이터 오브젝트가 RDF로 구현되었다면 다음 그림과 같은 개념적 관계의 매칭 프로세스를 거칠 수 있다.



<그림 3.6> 프로세스 오브젝트 데이터 매칭 개념적 관계도

매칭을 위한 유사도 측정 기술은 앞 절과 마찬가지로 자연언어처리를 통한 방법이나 그래프로 표현가능한 RDF, WSDL-S의 표현 체계들 사

이의 거리를 측정하는 여러 가지 기법을 활용할 수 있으며, 본 연구에서는 프로세스 데이터 오브젝트와 웹 서비스 네트워크 파라미터 사이를 연결하는 유사도 측정 함수로 잘 알려진 세 가지 함수를 소개한다.

· Exact Matching (EM): 데이터 오브젝트 o와 웹 서비스 파라미터 p의 이름 및 속성 정보에 대한 일치 여부를 통해 판단하는 기준으로, o와 p가 일치하는지 여부를 다음 형태의 함수 $EM(o.name==p.name)$로 표현된다.

· Cosine Similarity (CS): 데이터 오브젝트 o와 웹 서비스 파라미터 p의 매칭을 위하여 p와 관계된 WSDL에 기술된 정보에 포함된 단어와 데이터 오브젝트 o에 대한 기술 단어 정보를 이용하여 TF-IDF(Terfm Frequency - Inverse Document Frequency) 벡터 v_1, v_2를 생성하고, 이들의 코사인 값으로 유사도를 측정할 수 있다.

$$CS_{\cos} = \cos(\theta) = \frac{v_1 \bullet v_2}{||v_1|| \bullet ||v_2||}$$

이를 위해서는 데이터 오브젝트 o를 표현하는 기술하는 단어들에 대한 리스트 마련이 필요하고, 이를 기반으로 TF-IDF 벡터를 생성한다.

· WorldNet-Based Similarity (WNS): 동의어 및 유의어를 시맨틱 클래스로 정의하고 이에 대한 시맨틱 관계들을 구축한다. RDF 및 온톨로지와 같은 의미론적인 체계를 구축하고 이를 기반으로 데이터 오브젝트 및 웹 서비스 파라미터 사이의 거리를 측정한다. WorldNet은 영어 단어로 이루어진 대표적인 시맨틱 클래스들의 네트워크로 다양한 단어들 사이의 관계를 표현하는 체계이다. WorldNet 내에 있는 두 단어들 사이의 유사도를 측정하는 기술에 관하여 많은 연구들이 제시되었다. 그 중 데이터 오브젝트 o와 웹 서비스 파라미터 p 사이의 최소한의 공통 정보를 둘의 정보 합으로 나누는 Lin 유사도 (Fellbaum, 1998)가 많이 활용

된다. 데이터 오브젝트 o의 컨텐츠 c_o와 웹 서비스 파라미터 p의 컨텐츠를 c_p라고 정의했을 때, Lin 유사도는 다음과 같이 표현된다.

$$WNS_{LIN}(c_o, c_p) = \frac{2\log p_M(c_1, c_2)}{\log p(c_o) + \log p(c_p)}$$

여기서 $p(c)$는 컨텐츠 c의 인스턴스가 발견될 확률이고, $p_M(c_o, c_p)$는 컨텐츠 c_o와 c_p를 모두 포함하는 컨텐츠에 대한 최소 확률값을 의미한다.

3.2.3. 최적 웹 서비스 선택 및 WS-BPEL 작성

프로세스 데이터 오브젝트와 웹 서비스 네트워크의 파라미터들 사이의 의미론적 매칭이 끝나고 나면 최적화 엔진을 통해 주어진 네트워크에 대한 최적의 웹 서비스들을 찾는다. 프로세스 데이터 오브젝트는 현재 가용가능한 파라미터로 사용하고, 인스턴스의 부족한 정보(Underspecification)에 관한 상황을 보충하기 위한 대체 데이터 오브젝트를 웹 서비스 네트워크 임의의 파라미터 요소로부터 찾는다. 이를 모형화한 문제의 상세와 이를 풀 수 있는 알고리즘은 다음 장에서 다루도록 한다.

최적 웹 서비스 선택이 끝나고 나면 해당 웹 서비스들로 이루어진 WS-BPEL 프로세스를 작성하고, 작성된 프로세스는 워크플로우 프로세스는 늦은 모델링 모듈이 생성한 WS-BPEL 프로세스를 새로 할당하는 작업에 의해 실행된다.

4. 협업 프로세스 동적 구성을 위한 웹 서비스 컴포지션 문제

본 장에서는 3장에서 제시한 런타임 프로세스 동적 구성 프레임워크에서 동적으로 구성할 웹 서비스를 선택하는 최적화 문제를 정의한다. 그리고 해당 문제에 대한 계산 복잡도가 NP-Hard 임을 증명하고, 그 해법을 제시한다.

4.1. 네트워크 모형

늦은 모델링을 위해 서비스 저장소에 등록된 웹 서비스들을 바탕으로 웹 서비스 네트워크 모형을 구축하였다. 이 네트워크 모형에서 노드로 표현되는 요소는 웹 상에서 배포되는 웹 서비스와 이 웹 서비스를 실행시키기 위해 필요한 여러 메시지 타입(Message Type), 선후행 조건, 입출력 타입 등을 표현하는 데이터 오브젝트로 이루어진다. 이러한 데이터 오브젝트를 이후 리소스라 칭한다. 이러한 리소스는 웹 서비스 내의 오퍼레이션과 연결이 되고 실행 및 관련 비용 역시 오퍼레이션 단위로 각각 독립적으로 이루어진다. 본 절에서 제시하는 네트워크 모형에서는 이 오퍼레이션을 웹 서비스로 간주한다. 따라서 세 개의 오퍼레이션을 갖는 웹 서비스가 있다면, 각각의 오퍼레이션에 대한 웹 서비스 노드 세 개가 생성되고 할당된다. 오퍼레이션 사이의 통신하는 상황이 필요하더라도 이는 별도의 특수 리소스를 두 오퍼레이션 사이에 매개할 수 있으므로 모형이 야기하는 의미론적 손실이 없음은 쉽게 알 수 있다. 이후 웹 서비스의 기능을 직접 담당하는 오퍼레이션은 수리 모형에서 웹 서비스로 칭한다.

집합 S를 웹 서비스의 집합, R을 리소스의 집합이라고 하자. 그러면 각 웹 서비스 $s \in S$에 대하여 입력 리소스의 집합 $I_s \subseteq R$ 와 출력 리소스의 집합 $O_s \subseteq R$ 가 주어진다.

늦은 모델링을 위한 웹 서비스 컴포지션 문제는 초기 입력 리소스 $R_I \subseteq R$ 가 주어졌을 때, 목표 리소스 $R_T \subseteq R$ 을 얻기 위해 비용을 최소화하는 웹 서비스들을 선택하는 문제이다. 이 문제는 각 웹 서비스를 선택하여 실행하기 위해서는 입력으로 필요한 리소스가 미리 준비되어야 함을 가정한다. 즉, 이 문제는 현재 인스턴스에서 가용가능한 리소스 집합 R_I 가 있고 획득하고자 하는 목표 리소스 집합 R_T가 있을 때, 이 두 리소스

집합을 연결하는 최소비용의 서비스 집합을 찾는 것이다.

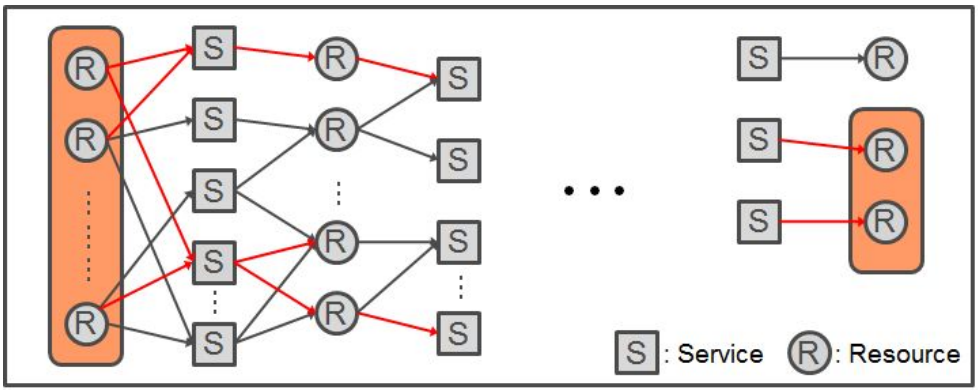
이러한 웹 서비스 컴포지션 문제는 다음과 같은 유향 그래프(Directed Graph) 모형으로 표현할 수 있다.

(ⅰ) $V=S \cup R$

(ⅱ) $(r,s) \in A$, if $r \in I_s$, and $s \in S$

(ⅲ) $(s,r) \in A$, if $r \in O_s$, and $s \in S$

위 정의에 의해 각 웹 서비스들은 각자의 입력 리소스와 출력 리소스를 갖게 되고, 노드와 노드 사이를 연결하는 호는 웹 서비스 노드와 리소스 노드 사이에서만 발생한다. 그리고 각 웹 서비스들이 입력과 출력으로 특정 리소스를 공유하게 되면 노드로 표현되는 이들 사이에 관계를 형성하게 된다. 입력 리소스가 필요없는 웹 서비스의 경우에는 더미 리소스 노드를 생성하여 이 리소스 노드를 해당 웹 서비스와 연결하는 방식으로 특별한 입력 리소스를 갖지 않는 경우에도 네트워크 상에서 표현할 수 있다. 웹 서비스와 리소스 사이의 관계를 통해 그려지는 네트워크의 모습은 다음 <그림 4.1>과 같다.



<그림 4.1> 웹 서비스-리소스 네트워크

그리고 네트워크에서 존재하는 비용은 다음과 같다. 임의의 웹 서비스 $s \in S$를 선택하여 실행할 때에는 실행 비용 c_s가 발생한다. 또한, 선택된 웹 서비스 s를 실행하여 얻은 리소스 O_s 중에서 다른 웹 서비스에 의해 사용되지 않은 임의의 리소스 r이 존재한다면, 필요없는 리소스를 생성함에 기인한 잔존 리소스 패널티 비용인 p_r이 발생한다.

이와 같은 조건 하에서 늦은 모델링을 위한 웹 서비스 컴포지션 문제의 가능해는 유향 그래프 G에서 다음을 만족하는 노드의 부분집합 $W \subseteq V$로 표현할 수 있다.

(i) If $v \in W \cap S$, $u \in W \cap R$, where $(u,v) \in A$ or $(v,u) \in A$.

(ii) If $v \in W \cap R$, there exists $u \in W \cap S$, such that $(u,v) \in A$.

(iii) $R_T \subseteq W$.

가능해 W에 포함된 서비스 노드 v가 선택이 되면, 이 서비스와 입력, 출력 관계를 가지는 리소스 u도 가능해 W에 포함이 된다. 그리고 가능해 W에 포함된 리소스 v가 존재하면, 이 리소스 v를 생성하는 임의의 서비스 u가 반드시 존재한다. 끝으로 협업 프로세스의 실행을 통해 얻고자 하는 목표 리소스에 해당하는 노드는 반드시 가능해 W 안에 포함되어야만 한다.

4.2. 최적화 모형

본 절에서는 늦은 모델링을 위한 웹 서비스 컴포지션 문제는 앞 절에서 정의한 네트워크 모형을 이용하여 다음과 같은 최적화 모형을 만들었다. 최적화 모형에 사용되는 파라미터와 의사결정변수는 다음과 같다.

Notation	
x_v	그래프의 노드 집합 V의 원소인 v가 선택되었는지 여부를 판단하는 이진 변수로서, 원소 v는 웹 서비스나 리소스가 될 수 있음.
y_{sr} (or y_{rs})	웹 서비스 집합 S⊆V의 원소인 s와 리소스 집합 R⊆V의 원소인 r 사이의 관계를 가지는지 여부를 판단하는 이진 변수로서, 웹 서비스가 선택이 되면 이와 관련된 리소스들이 입력 요소로 사용되고, 출력 요소로 생성되었는지 여부를 확인.
z_r	목표 리소스가 아닌 어떤 리소스 r이 웹 서비스에 의해 생성은 되었을 때, 이 리소스 r이 다른 웹 서비스에 의해 사용되었는지 여부를 판단하기 위한 이진 변수로서, 다른 서비스에 의해 사용되지 않은 경우에는 1, 사용된 경우에는 0의 값을 가짐.
$e_{s_1 s_2}$	웹 서비스들 사이의 관계를 나타내는 이진 변수로서 어떤 웹 서비스 s_1에 의해 만들어진 리소스가 다른 웹 서비스 s_2의 입력 요소로 사용되면 1, 그렇지 않은 경우 0의 값을 가짐.
d_v^{in}	그래프의 노드 집합 V의 원소인 v로 들어오는 호의 개수.
d_v^{out}	그래프의 노드 집합 V의 원소인 v에서 나가는 호의 개수.
c_s	웹 서비스 집합 S⊆V의 원소인 s의 실행비용.
p_r	리소스 집합 R⊆V의 원소인 r이 생성되었으나 잔존했을 때 발생하는 패널티 비용.
B	사용가능한 예산.

최적화 모형은 목표 리소스 R_T와 가용 리소스 R_I가 정의되어 있을 때, 가용 리소스 R_I 중 일부 또는 전부를 사용하여 모든 목표 리소스 R_T를 생성하기 위한 최소 비용의 웹 서비스 선택 문제로 정의할 수 있다.

이 문제의 목적함수는 협업 프로세스의 동적 구성을 위해 웹 서비스를 선택하고 호출하는데 들어가는 실행비용과 협업 프로세스를 실행하는 과정에서 생성이 되었으나 사용되지 않은 리소스의 잔존 패널티 비용의 합을 최소화하는 것을 목적으로 한다. 미리 정의한 표기법에 따라 목적함수식을 표현하면 식 (1)과 같다.

$$\min \sum_{s \in S} c_s x_s + \sum_{r \in R} p_r z_r \tag{1}$$

그리고 어떤 웹 서비스가 선택이 되면, 해당 웹 서비스를 실행하는데 필요한 입력 리소스와 출력 리소스가 선택이 되어야만 한다. 앞의 네트워크 모형에서 노드와 노드 사이를 잇는 호는 웹 서비스의 입출력 관계를 기반으로 만들어졌음을 설명했다. 따라서 이 웹 서비스가 사용하는 입력 리소스의 개수만큼의 자신으로 들어오는 호를 선택하게 되면, 정의된 모든 입력 리소스를 선택하는 것이 되고, 출력 리소스의 개수만큼의 나가는 호가 있어야 한다고 정의하면 모든 출력 리소스들을 선택할 수 있게 된다. 웹 서비스가 선택되면 해당 웹 서비스의 입력 리소스 개수만큼 들어오는 호를 허용하는 제약식은 식 (2)와 같고, 출력 리소스 개수만큼의 나가는 호를 허용하는 제약식은 식 (3)으로 표현할 수 있다.

$$\sum_{(r,s) \in A} y_{rs} = d_s^{in} x_s \quad , \forall s \in S \tag{2}$$

$$\sum_{(s,r) \in A} y_{sr} = d_s^{out} x_s \quad , \forall s \in S \tag{3}$$

다음은 리소스에 관한 제약식이다. 먼저 가용 리소스가 아닌 어떤 리소스가 생성이 되었다면, 해당 리소스는 어떤 웹 서비스에 의해 생성이 된 것이다. 따라서 이 리소스를 만드는 웹 서비스 중 적어도 하나 이상이 선택되었음을 알 수 있고, 이를 수식으로 표현하면 식 (4)와 같다. 또한 목표 리소스가 아닌 어떤 리소스가 웹 서비스에 의해 사용이 되기 위해서는 해당 리소스가 미리 준비되어 있어야 함을 나타내는 제약식은 식 (5)와 같다.

$$\sum_{(s,r)\in A} y_{sr} \geqq x_r, \quad \forall r \in R \setminus R_I \tag{4}$$

$$\sum_{(r,s)\in A} y_{rs} \leqq M x_r \quad , \forall r \in R \setminus R_T \tag{5}$$

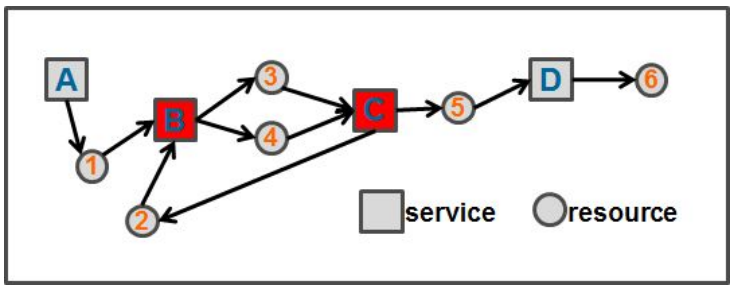
동적 협업 프로세스 구성을 위해 호출한 웹 서비스들로 인해 생성된 리소스 중 다른 웹 서비스에 의해 사용되지 않고, 남아있는 리소스를 판단하기 위해 이진 의사결정변수 z_r을 사용하여 식 (6)와 같은 제약을 형성한다.

$$x_r - \sum_{(r,s)\in A} y_{rs} \leqq z_r \ , \quad \forall r \in R \tag{6}$$

식 (6)에서 어떤 리소스 r이 생성이 되면 의사결정변수 x_r은 1의 값을 가질 것이고, 이 r을 입력 리소스로 사용하는 모든 웹 서비스들 중 하나라도 호출이 되어 실행된다면 $\sum_{(r,s)\in A} y_{rs}$은 1이상의 값을 갖게 된다. 따라서 $\sum_{s\in S} c_s x_s + \sum_{r\in R} p_r z_r$을 최소화하려는 목적함수에 의해 이진 변수인 z_r은 0의 값을 가져야만 한다. 즉, 생성이 된 리소스가 사용이 된 경우에는 잔존 리소스에 대한 패널티를 받지 않는 것이다. 반면에 만약 리소스 r이 생성이 됐지만, 다른 웹 서비스에 의해 사용되지 않는다면, $\sum_{(r,s)\in A} y_{rs}$은 0

의 값을 갖게 되고, 식 (6)의 부등제약을 만족시키기 위해서 이진 변수인 z_r의 값은 1이 되어야 한다.

앞에서 정의한 네트워크 모형에 의해, 각기 다른 비용을 가지는 웹 서비스들은 공유 및 경쟁하는 리소스를 통하여 네트워크를 형성한다. 하지만 이 문제와 같은 0-1 정수계획 모형의 경우, 일반적인 Branch and Bound로 해를 찾을 때 다음 <그림 4.2>의 웹 서비스 B와 C처럼 사이클을 형성하는 해가 나타날 수 있다. Branch and Bound는 각각의 이진 의사결정변수를 0 또는 1의 값의 경우로 나누고 각각의 분기 경우에 대한 모든 해공간을 탐색한다. 따라서 만약 서비스에 대한 의사결정변수 x_c와 x_b의 결정 순서에서 x_b가 선택된 분기에서는 리소스 3과 4는 사용가능한 형태로 인식이 될 것이고, x_c는 입력 리소스가 존재하므로 실행가능하다. 이 분기에서 최적해가 존재하는 경우에는 이 해는 사이클을 포함하기 때문에 사용할 수 없다.



<그림 4.2> 사이클이 존재하는 경우

따라서 이러한 사이클을 포함하는 해를 제거하기 위한 제약식이 필요하다. 웹 서비스 집합 S에 대한 임의의 부분집합 N이 있을 때, 이들 각각이 특정 리소스를 매개로 연결되었는지 여부를 판단하는 이진 변수 e_{ij}를 정의한다. 그리고 이를 이용하여 집합 N에 포함된 노드의 e_{ij}의 합이 집합 N에 포함된 노드의 개수와 같으면 사이클이 발생하므로, 이 합이 집합 N의 원소 개수보다는 무조건 작은 값을 가지도록 제약식을 만든

다. 이를 표현하면 식 (7)과 같다.

$$\sum_{i \in N} \sum_{j \in N} e_{ij} \leqq |N| - 1 \text{ for } N \subset S,\ 2 \leqq |N| \leqq |S| - 1 \qquad (7)$$

예산 제약은 웹 서비스를 호출하는데 필요한 실행비용과 협업 프로세스 구성 과정에서 불필요하게 남게 되는 중간 리소스의 잔재비용의 합에 대한 제약이다. 식 (8)은 주어진 예산 B 내에서 실행되어야 한다는 제약식이다.

$$\sum_{s \in S} c_s x_s + \sum_{r \in R} p_r z_r \leqq B \qquad (8)$$

앞에서 정의한 바와 같이 가용 리소스는 협업 프로세스 구성을 위하여 현재 사용할 수 있는 리소스이며, 목표 리소스는 협업 프로세스를 통해서 얻고자 하는 대상이다. 따라서 현재 인스턴스에 주어진 가용 리소스는 전체 혹은 일부 사용할 수 있으며, 목표 리소스는 반드시 생성하여야만 한다. 가용 리소스에 대한 제약식은 식 (9)와 같고, 목표 리소스에 대한 제약식은 식 (10)과 같다.

$$\sum_{r \in R_I} x_r \geqq 1 \qquad (9)$$

$$x_r = 1\ ,\ \forall r \in R_T \qquad (10)$$

끝으로 어떤 웹 서비스나 리소스가 선택되었는지 여부를 판단하는 의사결정변수 x_s와 잔존 리소스가 생성되었는지 여부를 판단하는 의사결정변수 z_r이 0 또는 1의 값을 갖는다는 제약은 다음과 같다.

$$x_v \in \{0,1\}\ ,\ \forall v \in V \tag{11}$$

$$z_r \in \{0,1\}\ ,\ \forall r \in R \tag{12}$$

4.3. 계산 복잡도

본 절에서는 앞 절에서 제시한 최적화 모형의 계산 복잡도를 증명한다. 여기서 늦은 모델링을 위한 웹 서비스 컴포지션 문제는 하나의 계층으로 구성되는 간단한 인스턴스가 NP-hard 임을 보일 것이다. 이의 증명은 본 문제가 Set Cover 문제로부터의 다항시간 변환(Polynomial Time Reduction)이 됨으로써 이루어진다.

(1) Set Cover 문제의 정의 (Vazirani, 2000)

U={1, 2, ... , n}의 부분집합인 S_i, i=1, 2, ..., m은 $c(S_i) \in R_+$만큼의 크기를 가진다. 임의의 양의 실수 $K \in R_+$가 주어졌을 때, 다음의 속성을 만족하는 S_i의 모음(Collection)인 C가 존재한다.

$$\cup_{S \in C} S_i = U$$

$$\sum_{S \in C} c(S) \leqq K$$

(2) 증명

증명은 Set Cover의 인스턴스가 참일 때, 본 문제의 인스턴스 역시 참임을 보이고, 그 역도 성립함을 통해 이루어진다. (1)에서 보인 정의에 따라 Set Cover 문제의 인스턴스로부터 본 문제의 인스턴스를 다음의 과정을 통해 생성한다.

- 늦은 모델링을 위한 웹 서비스 컴포지션 문제는 m개의 서비스와 하나의 입력 리소스가 존재하고, 이들 서비스는 모두 하나의 입력 리소스를 공유한다. - U에 속한 각 원소에 해당하는 목표 리소스가 n개 존재하고, 각각의 목표 리소스는 S_i에 속한 원소에 해당하는 서비스를 실행할 때 생성된다. - 각 서비스의 실행비용은 $c(S_i)$이고, 각 리소스에 대한 잔존 패널티 비용은 0으로 고정된다.

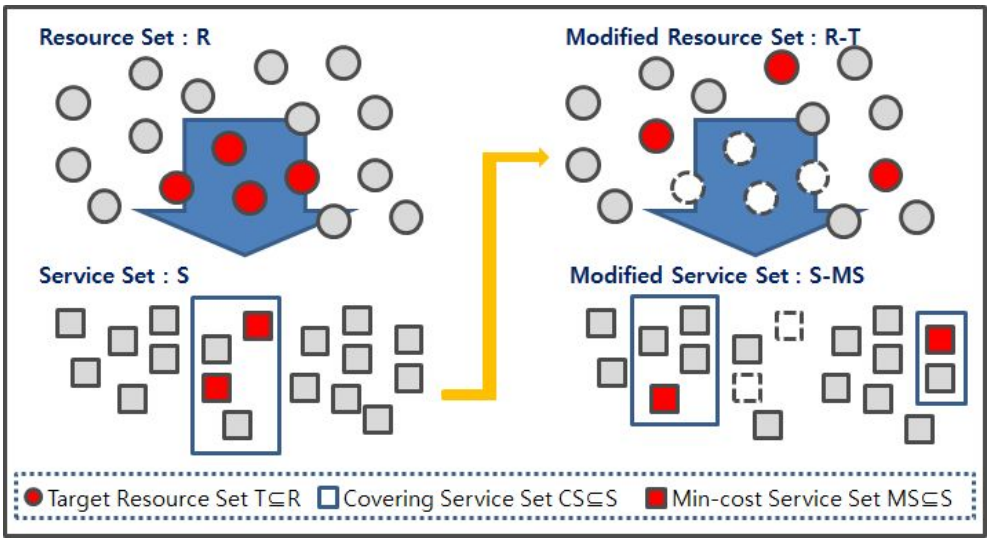
만약 Set Cover의 인스턴스가 위 정의를 만족하는 특수한 인스턴스일 때, S_i의 모음인 C를 실행을 하게 되면 모든 U의 원소가 생성이 되고, 이에 대응되는 웹 서비스 집합을 선택을 하게 되면 모든 목표 리소스를 다 생성하는 것으로 볼 수 있다. 그리고 두 인스턴스의 비용 역시 동일하다. 반대 방향도 비슷한 방법으로 증명할 수 있다.

4.4. 웹 서비스 선택을 위한 휴리스틱

이전 4.3절에서 증명했듯이, 늦은 모델링을 위한 웹 서비스 컴포지션 문제는 높은 계산 복잡도를 가져 다항 연산 시간 내에 풀 수 없는 문제이다. 따라서 본 절에서는 최적해를 보장하지는 못하지만 이에 근접하는 해를 제공하는 두 가지 방식의 휴리스틱을 제안한다.

4.4.1. 후방 검색 (Backward Search) 알고리즘

후방 검색 알고리즘은 늦은 모델링을 위한 웹 서비스 컴포지션 문제에서 가장 직관적으로 생각해낼 수 있는 방법이다. 후방 검색 알고리즘의 기본적인 원리는 목표 리소스를 만들어내는 최소 비용의 웹 서비스들을 선택하고, 선택된 웹 서비스들의 입력 리소스를 목표 리소스로 갱신하여 이들을 만들어내는 웹 서비스들을 다시 선택하는 과정을 반복한다. 그리고 이러한 과정은 <그림 4.3>과 같이 필요로 하는 입력 리소스의 집합이 가용 리소스의 부분집합이 될 때까지 반복 수행된다.



<그림 4.3> 후방 검색 알고리즘의 개요

하지만 후방 검색을 직접 적용할 시, 이 알고리즘 역시 높은 계산 복잡도를 갖게 된다. 그 이유는 특정 리소스를 만드는 최소 비용의 웹 서비스 집합을 찾는 것 역시 Set Cover 문제로부터 다항시간 변환이 되는 높은 계산 복잡도를 갖는 문제이기 때문이다.

따라서 특정 목표 리소스를 생성할 수 있는 웹 서비스들을 대상으로 평가 지표를 정의하고, 이 지표에 따라 웹 서비스에 대한 점수를 매긴다. 그리고 이 점수가 높은 웹 서비스에 우선 순위를 부여해 특정 목표 리소스를 만드는 웹 서비스로 선택하는 기준으로 활용한다. 후방 검색을 위해 정의된 웹 서비스 평가 지표(WEM : Web Service Evaluation Metric)는 다음 수식 (11)과 같다.

$$WEM(x_j) = \frac{n(x_j)}{ec(x_j) + pc(x_j)} \tag{11}$$

식 (11)의 WEM(x_j)의 값은 웹 서비스 x_i를 실행하는데 필요한 비용 ec(x_j)와 이 웹 서비스가 생성하는 목표 리소스를 제외한 리소스들의 잔존 패널티 비용 pc(x_j)의 합에 반비례하고, 이 웹 서비스를 실행하여 생성할 수 있는 목표 리소스의 개수 n(x_j)에는 비례함을 보여준다. 따라서 목표 리소스를 많이 만들면서, 적은 비용을 갖는 웹 서비스를 우선적으로 선택할 수 있다. 그리고 한 번 선택된 웹 서비스는 선택할 수 있는 웹 서비스 집합에서 제외하여 사이클을 형성하는 것을 막는다. 특정 목표 리소스를 생성하는 웹 서비스들을 찾기 위하여 잘 알려진 Greedy Set Cover Algorithm (Vairani, 2000)을 수정해서 적용하였으며, 이에 대한 의사코드(Pseudo Code)는 다음 <그림 4.4>와 같다.

INPUT	O : 출력 리소스(Output Resource) 집합 ec(s) : 웹 서비스 s의 실행비용 pc(s) : 웹 서비스 s가 생성하는 불필요 리소스에 대한 패널티 비용
OUTPUT	COVER : 출력 리소스 생성을 위해 선택된 웹 서비스 집합 COST : 선택된 웹 서비스 집합 COVER의 총비용으로 ec(s)+pc(s)
FUNCTION	findService(r) : 리소스 r을 생성하는 웹 서비스 검색 함수 findOutput(s) : 웹 서비스 s가 생성하는 리소스 검색 함수 calWEM(s) : 웹 서비스 s에 대한 WEM 값 계산 함수

```
BEGIN
COVER← ∅, COST = 0
WHILE(O⊈∅)
        Every resource r in O
        S← FindService(r)
        For every s in S
                calWEM(s)
                b← Select the web service which has biggest
                        WEM value
        For END
        S← S - {b}
        COVER← COVER∪{b}
        COST = COST + ec(b)+pc(b)
        O← O - {r}
WHILE END
RETURN COVER, COST
END
```

<그림 4.4> 특정 목표 리소스 생성을 위한 Greedy Set Cover 의사코드

<그림 4.4>의 Greedy Set Cover 알고리즘을 이용하여 특정 목표 리소스를 생성하는 웹 서비스 집합을 찾은 다음, 이 웹 서비스들이 필요로 하는 입력 리소스를 다시 찾고, 이러한 과정을 모든 입력 리소스 집합이 가용 리소스의 부분집합이 될 때까지 진행한다. Greedy Set Cover 알고

리즘에서 선택된 웹 서비스들의 입력 리소스들은 잔존 패널티 비용을 0으로 설정한다. Greedy Set Cover 알고리즘을 포함한 후방 검색 알고리즘은 다음 <그림 4.5>와 같다.

INPUT	AR : 가용 리소스(Available Resource) 집합 TR : 목표 리소스(Target Resource) 집합
OUTPUT	COVER : 목표 리소스 생성을 위해 선택된 웹 서비스 집합 COST : 선택된 웹 서비스 집합 COVER의 총비용
FUNCTION	findCoverSet(O) : 주어진 출력 리소스 O에 대하여 Greedy Set Cover 알고리즘을 통해 반환된 웹 서비스 집합 findCoverCost(O) : 주어진 출력 리소스 O에 대하여 Greedy Set Cover 알고리즘을 통해 반환된 비용 findInput(S) : 웹 서비스 집합 S에 포함된 모든 웹 서비스를 실행하기 위해 필요한 리소스 검색 함수

```
BEGIN
OUTPUT← TR
COVER← ∅ , COST = 0
WHILE( OUTPUT⊈AR )
        D← findCoverSet(OUTPUT)
        c = findCoverCost(OUTPUT)
        COVER← COVER∪D
        COST← COST + c
        OUTPUT← findInput(D)
WHILE END
RETURN COVER, COST
END
```

<그림 4.5> 후방 검색 의사코드

후방 검색 알고리즘은 최적해를 보장하지 않는다. 그 이유는 두 가지가 있는데, 첫째, 후방 검색 알고리즘은 주어진 목표 리소스 집합에서 출발하여 해당 리소스를 만드는 최소 비용의 집합을 단계적으로 반복하는

방식을 취하는데, 각 단계별 선택된 최소 비용의 집합은 이후 단계의 비용을 전혀 고려하지 못하기 때문이다. 또한 특정 리소스를 만드는 최소 비용의 웹 서비스 집합을 찾는 것 역시 Set Cover 문제로부터 다항시간 변환이 되는 높은 계산 복잡도를 갖는 문제이다. 따라서 후방 검색 알고리즘에서는 이에 대한 연산 시간을 줄이고자, 특정 리소스를 만드는 모든 웹 서비스에 대하여 일괄적으로 WEM을 계산하였다.

4.4.2. 유방향 최소신장트리(Directed Minimum Spanning Tree)를 이용한 알고리즘

본 문제는 네트워크가 주어져 있을 때, 임의의 가용 리소스를 이용하여 최종적으로 목표 리소스를 만들기 위해 필요한 웹 서비스를 최소비용으로 찾는 문제로 볼 수 있다. 따라서 본 논문에서는 주어진 늦은 모델링을 위한 웹 서비스 컴포지션 문제를 풀기 위하여 유방향 최소신장트리(Directed Minimum Spanning Tree) 알고리즘을 활용한다.

트리 형태의 해를 찾는 알고리즘을 사용하는 이유는 우선 경로 형태의 해는 모든 입력 리소스를 반영할 수 없기 때문이다. 예를 들어 어떤 가용 리소스 1을 입력 리소스로 사용하는 웹 서비스 A와 B가 있고, 웹 서비스 A는 리소스 2, 3을 B는 리소스 4, 5를 생성한다. 그리고 웹 서비스 C는 리소스 3와 4를 입력 리소스로 사용하여 리소스 6를 생성하는데, 이 리소스 6가 목표 리소스로 주어진 상황을 생각해보자. 일반적인 경로 알고리즘을 이용하여 해를 찾을 시, 웹 서비스 A와 B 둘 중 하나는 선택이 되지 않는다. 즉 병렬적인 형태로 결합이 되어 있는 웹 서비스, 리소스 관계를 반영하기 힘들다.

다른 이유는 리소스 노드에 부여되는 잔존 패널티 비용의 속성 때문이다. 앞의 정의에 따라 잔존 패널티 비용은 리소스 노드로 들어오는 호는 존재하지만, 나가는 호가 존재하지 않을 경우에 발생한다. 따라서 가용 리소스와 목표 리소스를 제외한 경로 내의 중간 리소스 노드들은 모두 해당 노드의 패널티 비용이 0이 되버리기 때문이다. 하지만 경로 내에 있는 웹 서비스들의 실행으로 인하여 불필요하게 생성된 리소스는 존재할 수 있으므로, 이들의 잔존 패널티 비용을 경로의 비용 평가에 반영할 수 있는 방법 역시 필요하다.

신장트리(Spanning Tree)는 주어진 그래프에서 트리이고, 모든 노드를 포함하는 서브그래프로 정의할 수 있다. 어떤 그래프가 주어졌을 때, 이 그래프 내에는 여러 개의 신장트리가 존재할 수 있고, 각각의 호에는 비용과 같이 작을수록 좋은 가중치를 부여할 수 있다. 최소신장트리는 주어진 그래프 내에 존재하는 여러 신장트리 중 호의 비용합이 가장 작은 신장트리를 의미한다. 유방향 최소신장트리 문제를 푸는 알고리즘으로는 Edmonds 알고리즘(혹은 Chu-Liu/Edmonds 알고리즘)이 널리 알려져 있다 (Vazirani, 2000). 본 연구에서는 Edmonds 알고리즘을 활용하여 주어진 유방향 네트워크의 모든 노드를 포함하는 최소신장트리를 만들고, 이 최소신장트리에서 불필요한 노드를 제거하는 방식으로 해를 찾는 방식을 취한다.

주어진 늦은 모델링을 위한 웹 서비스 컴포지션 문제는 노드에 비용이 부여된 형태이기 때문에 최소신장트리 알고리즘을 적용하기 위해서는 주어진 문제의 네트워크를 호에 비용을 부여한 형태로 변형해야 한다. 따라서 다음의 규칙에 따라 노드의 비용을 호로 이동한다.

· Rule 1. *어떤 웹 서비스 노드 j가 입력 리소스로 사용하는 집합 I_j가 있을 때, 이 I_j를 하나의 노드로 네트워크에서 하나의 리소스로 간주하고, 이를 웹 서비스 노드 j와 하나의 호로 연결한다.*

· Rule 2. *어떤 웹 서비스 노드 i의 출력 리소스 집합 O_i의 부분집합이 웹 서비스 노드 j의 입력 리소스 집합 I_j에 포함되면, 웹 서비스 노드 i와 I_j를 연결하는 호를 생성한다.*

Rule 1을 통해 네트워크의 노드 집합을 변형시키게 되면, 두 개 이상의 입력 리소스를 필요로 하는 웹 서비스를 위한 새로운 리소스 노드를 네트워크에 생성하게 된다. 또한 모든 웹 서비스는 하나의 입력 리소스를 받는 형태로 변형된다. 그리고 Rule 2는 Rule 1로 생성된 입력 리소스와 이들을 생성하는 웹 서비스 노드를 연결하는 호를 만들게 된다. 하

나의 입력 리소스 형태로 네트워크를 수정하기 위해 임의로 생성된 노드는 별도의 멀티노드 집합 M⊆V로 구분한다.

・Rule 3. *Rule 2로 생성된 호의 비용 $c(i, I_j)$은 리소스 집합 $O_i \backslash I_j$ 의 원소에 해당하는 리소스의 잔존 패널티 비용 합이다.*

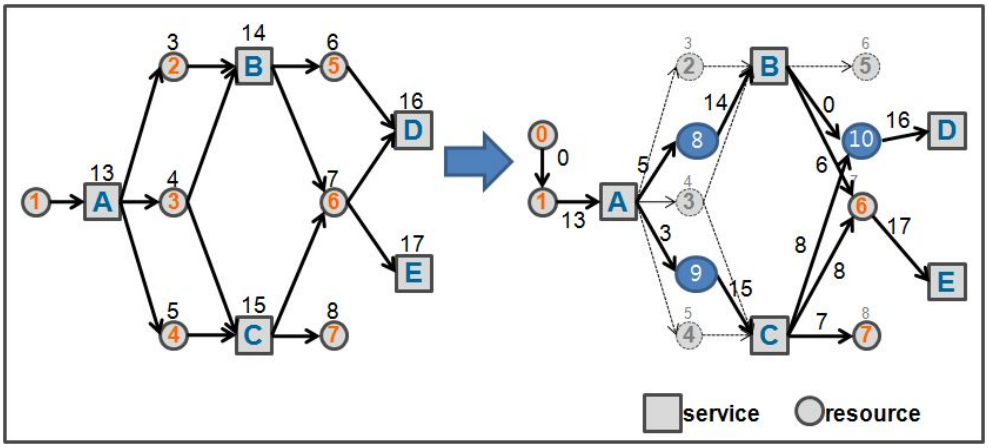
・Rule 4. *어떤 리소스 i를 사용하는 서비스 j가 있고 이 서비스 노드 j의 실행 비용을 c_j라 하면, 호의 비용 $c(i, j)$는 c_j이다.*

・Rule 5. *모든 가용 리소스로 향하는 호를 가지는 임의의 더미 노드 (Dummy Node)를 생성하고, 이 더미 노드와 가용 리소스 사이의 비용은 0이다.*

Rule 3은 웹 서비스에서 리소스로 가는 호의 비용을 결정한다. Rule 3으로 인하여 웹 서비스가 생성한 리소스 중 특정 리소스만 임의의 서비스에 의해 사용될 경우, 사용되지 않은 리소스의 모든 잔존 패널티 비용을 호로 옮겨오게 된다. 잔존 리소스와 관련된 비용을 줄이는 방향으로 해를 찾도록 유도할 수 있다. 그리고 Rule 4는 리소스에서 웹 서비스로 가는 호의 비용은 해당 웹 서비스의 실행 비용으로 설정하기 위하여 만든 규칙이다.

마지막으로 Rule 5는 Edmonds 알고리즘에서 루트 노드를 설정하기 위하여 더미 노드를 생성하는 규칙이다. Edmonds 알고리즘은 루트 노드가 아닌 다른 모든 노드로 들어오는 최소비용인 호를 하나씩 선택하고, 특정 노드들이 사이클을 형성할 경우, 이를 없앨 수 있는 다른 최소비용의 호를 사이클에 포함되지 않은 노드로부터 가져오는 방식을 취한다. 하지만 주어진 가용 리소스는 두 개 이상이 될 수 있기 때문에 루트 노드 설정을 위한 임의의 더미 노드를 생성한다. 또한 이 더미 노드로부터 가용 리소스 노드로 가는 호의 비용은 0이므로 알고리즘 수행 결과 최소신장트리의 해 안에 반드시 포함될 수 있도록 하였다. 위에서 정의한 규

칙들에 따라 네트워크의 비용을 호로 옮긴 형태의 예는 다음 <그림 4.6>과 같다.



<그림 4.6> 최소신장트리 알고리즘 적용을 위한 네트워크 수정 예

위 그림에서 웹 서비스 A는 리소스 2, 3, 4를 생성하고, 웹 서비스 B는 이들 중 2, 3을 입력 리소스로 사용한다. 따라서 웹 서비스 B는 임의의 리소스 집합을 매개로 웹 서비스 A와 연결이 되어 있음을 알 수 있으며, 리소스 2, 3을 모두 포함하는 의미를 가지는 임의의 리소스 8을 네트워크에 생성한다. 이후 웹 서비스 노드 A에서 리소스 노드 8로 향하는 호와 리소스 노드 8에서부터 웹 서비스 노드 B로 향하는 호를 생성하여 입출력 관계를 완성한다.

그리고 호의 비용을 결정하는 경우는 다음과 같다. 먼저 웹 서비스 노드에서 리소스 노드로 향하는 호의 비용을 결정하는 예를 보자. <그림 4.6>의 웹 서비스 A에서 리소스 노드 8로 가는 비용은 리소스 2, 3을 제외한 나머지 A의 출력 리소스의 잔존 패널티 비용의 합으로 결정한다. 따라서 위 예의 c(A, 8)의 비용은 리소스 5의 잔존 패널티 비용인 5로 결정되고, 마찬가지의 방식으로 c(A, 9)는 3, c(B, 10)은 0의 값을 갖는다. 끝으로 리소스 노드에서 웹 서비스 노드로 향하는 호의 비용은 호가

향하는 웹 서비스의 실행비용을 갖도록 한다. 따라서 호의 비용 c(1, A)는 웹 서비스 A의 실행비용인 13이 되고, 같은 방법으로 c(8, B)는 14, c(9, C)는 15, c(10, D)은 16, c(6, E)은 17로 결정한다.

정의한 규칙에 의거하여 네트워크를 변형한 후, Edmonds 알고리즘을 이용하여 최소신장트리를 구한다. 알고리즘에 의해 생성된 해는 트리의 정의에 의해 사이클을 가지지 않는다. 본 문제에 적용한 Edmonds 알고리즘의 의사코드는 다음 <그림 4.7>과 같다.

INPUT	D(V,A) : 유방향 그래프 r : 루트 노드
OUTPUT	D(V,E) : 유방향 최소신장트리
FUNCTION	findMinCostArc(V) : 주어진 노드 V로 들어오는 호 중 최소비용인 호를 찾는 함수 findCycle(A) : 주어진 호 집합 A에 사이클이 존재하는지 여부를 판단하는 함수

```
1.  BEGIN
2.  A← ∅
3.  FOR each node v∈V, except root node r
4.      A← A∪findMinCostArc( V\{r} )
5.  FOR END
6.  FindCycle(A)
7.  IF no cycle is formed,
8.      D(V,A) is a Minimum Spanning Tree go to Line 21
9.  IF cycle is formed,
10.     K← the nodes in the cycle
11.     temp = ∞, T← ∅
12.     FOR each node k∈K, i∉K
13.         // x(k),k is the arc in the cycle which enters k.
14.         IF (temp ≧ c(i, k) - c( (x(k), k) )
15.             temp = c(i, k) - c( (x(k), k) )
16.             T← { (i, k) }
17.         END IF
18.     END FOR
19.     A← A∪T-{ (x(k), k) }
20. END IF
21. RETURN G(V, A)
22. END
```

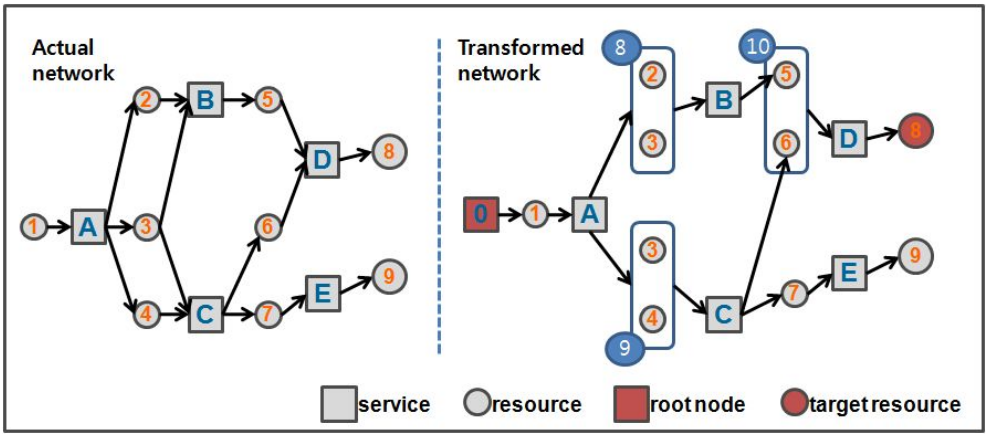
<그림 4.7> Edmonds 알고리즘

Edmonds 알고리즘으로 생성된 해는 주어진 네트워크 안에 있는 모든 노드들을 포함하는 트리이다. 주어진 최소신장트리에서 다음 세 단계의 후처리를 거친다.

· 1단계. 최소신장트리의 경로 웹 서비스 선택

Edmonds 알고리즘으로 얻어진 최소신장트리 G(V, A)에서 모든 웹 서비스, 리소스 노드는 자신으로 들어오는 하나 이상의 호가 주어진다. 그리고 Edmonds 알고리즘의 성질에 의하여 이들 모든 노드들은 루트 노드에서부터 출발하여 도달할 수 있다. 따라서 목표 리소스에서 출발하여 루트 노드로 도착하는 역방향의 경로 역시 반드시 존재한다. 각 목표 리소스에 대하여 이와 같은 역방향 경로를 추적하고, 이 경로 위에 존재하는 웹 서비스 노드들을 $COVER_1$ 집합에 저장한다.

1단계로 구해진 $COVER_1$ 집합은 가능해를 보장하지 않기 때문에 이를 보완하는 웹 서비스를 $COVER_1$ 집합으로 추가하는 과정이 필요하다. 1단계 결과로 얻은 $COVER_1$ 집합이 가능해를 보장하지 않는 경우는 다음 <그림 4.8>과 같다.



<그림 4.8> 1단계 $COVER_1$ 집합이 가능해를 보장하지 않는 경우 예

<그림 4.8>의 우측의 그래프를 변형한 형태의 네트워크에서 찾은 최소신장트리로, 좌측의 그래프를 해당 네트워크의 원 네트워크의 최소신장트리라고 하자. 그러면 1단계 처리 결과, 목표 리소스 8을 만들기 위한 경로 웹 서비스로 {A, B, D} 또는 {A, C, D}가 선택된다. 하지만 목표 리소스 8을 만드는 웹 서비스 D의 경우 리소스 5는 웹 서비스 B의 출력물로부터, 리소스 6은 웹 서비스 C의 출력물로 획득한 것이다. 따라서 경로를 추적하는 과정에서 멀티노드 M이 포함된 경우에는 이를 위한 웹 서비스 노드를 추가해야 한다.

· 2단계. 가능해를 위한 최소 비용 웹 서비스 추가

이를 위하여 2단계에서는 1단계에서 얻은 경로에서 멀티노드가 포함된 경우, 멀티노드로 향하는 호를 갖는 웹 서비스의 출력 리소스를 검색하여 M\O가 공집합이 아닐 시 Edmonds 알고리즘을 반복수행한다. 이때 Edmonds 알고리즘에 입력으로 사용되는 그래프는 경로 상에서 해당 서비스 실행에 영향을 미치는 웹 서비스들과 그들과 관련을 맺는 리소스로 진행한다.

따라서 Edmonds 알고리즘의 입력인 G(V, A)에서 $COVER_1$에 포함된 모든 웹 서비스 노드를 제거한 $V'=V\backslash COVER_1$로 갱신되고, 호의 집합 A 역시 제거된 웹 서비스 노드와 관련이 있는 호를 제거한 나머지 호들을 알고리즘의 입력 파라미터로 사용된다. 이들 노드와 호를 제거하여 Edmonds 알고리즘을 사용하는 이유는 탐색 공간을 줄여 계산의 효율을 높이려는 의도와 동시에 생성된 해에서 사이클이 발생하는 것을 막을 수 있다. 목표 리소스 R_T는 멀티노드 안에 포함된 웹 서비스들의 경로 상에 존재하지 않는 리소스로 설정한다.

· 3단계. 비용 계산

3단계에서는 1단계와 2단계로 구한 $COVER_1$과 $COVER_2$의 합집합을 구하여 필요한 웹 서비스를 확정짓고, COVER에 있는 모든 웹 서비스의 입력 노드 집합 I_{cover}와 출력 노드 집합 O_{cover}를 구한다. 그 다음 모든 COVER 집합에 있는 웹 서비스의 실행 비용 $C = \sum_{i \in COVER} c_i$ 과 $O_{cover} \backslash (I_{cover} \cup R_T)$ 에 해당하는 모든 리소스 노드의 잔존 패널티 비용 $P = \sum_{r \in O_{cover} \backslash (I_{cover} \cup R_T)} p_r$ 을 더해 총비용 COST=C+P 를 얻는다.

5. 웹 서비스 네트워크에 따른 협업 프로세스 구성 휴리스틱 성능 실험

본 장에서는 4장에서 다룬 협업 프로세스 동적 구성을 위한 웹 서비스 선택 문제와 제안한 두 개의 휴리스틱의 가해성을 확인하기 위한 실험을 설명한다. 본 연구의 대상은 런타임 프로세스의 인스턴스이기 때문에 웹 서비스 선택을 위한 알고리즘의 연산 시간이 중요하다. 또한 웹 서비스의 선택과 관련된 비용 요소도 고려해야 한다. 이에 대한 검증을 위해 웹 서비스를 선택하는데 영향을 미치는 웹 서비스 네트워크의 크기와 복잡한 정도에 따라 알고리즘의 성능을 비교한다. 비교에 사용되는 알고리즘은 최적해를 보장하지만 높은 연산 시간을 가지는 Branch and Bound 알고리즘과 최적해를 보장하지는 못하지만 계산 복잡도 면에서의 개선을 한 두 휴리스틱 알고리즘의 수행 시간과 모델링된 비용의 차이를 비교한다.

5.1. 실험 설계

(1) 네트워크 생성

알고리즘의 성능 비교를 위해 웹 서비스 네트워크의 크기와 복잡도를 변경하며 실험을 수행하였다. 네트워크의 크기에 따른 알고리즘 성능 비교를 위해 노드로 표현되는 웹 서비스 및 리소스 노드 개수에 따른 결과를 비교한다. 또한 네트워크의 복잡한 정도를 표현하기 위해 웹 서비스가 관련을 맺는 리소스의 수를 변화시켜 웹 서비스 네트워크의 복잡도에 따른 알고리즘 성능 비교를 수행한다. Kil *et al.* (Kil *et al.*, 2009)은 웹 서비스 공용 저장소인 XMethods.org 와 BindingPoint.com 등에서 WSDL을 구하고, 이를 통하여 웹 서비스 네트워크에 존재하는 오퍼레이션과 파라미터의 수의 관계에 대한 분석을 수행하였다. 대부분의 오퍼레이션이 두 개의 입출력 파라미터를 가진다는 점에 착안하여 평균 하나의 웹 서비스는 두 개의 독립적인 리소스와 관련을 맺는다는 전제하에, 네트워크에 존재하는 독립적인 리소스 수는 웹 서비스 수의 두 배로 가정한다.

- 웹 서비스와 리소스 노드의 수

실험에서 다루는 웹 서비스 네트워크에 표현되는 웹 서비스와 리소스의 수는 (25, 50), (50, 100), (100, 200), (200, 400)로 총 네 가지의 경우를 고려하였다.

- 네트워크 복잡도

네트워크의 복잡한 정도를 판단하는 기준 중 하나는 네트워크에 존재하는 호의 개수이다. 네트워크의 복잡한 정도를 반영하기 위하여 웹 서비스가 관계를 맺는 입출력 리소스의 개수를 변경하여 네트워크를 완성하였다. 실험에서 다루는 웹 서비스 네트워크는 입출력 리소스의 평균

연결수가 3(2~4)인 경우, 6(5~7)인 경우, 9(8~10)인 경우로 나누어 고려하였다. 주어진 개수의 웹 서비스 노드와 리소스 노드가 있을 때, 각 웹 서비스는 주어진 조건의 연결수에 맞춰 리소스를 임의로 선택한다.

(2) 비용 설정

모형에서 설정한 두 종류의 비용인 웹 서비스 실행 비용과 리소스 잔존 패널티 비용은 0~1 사이의 값으로 랜덤하게 생성하여 할당하였다. 웹 서비스 QoS의 관점에서 웹 서비스의 실행 비용은 작동을 위해 지불해야 하는 비용 뿐만 아니라 신뢰도, 정확도 등의 속성을 포함하는 개념이다. 실험에서 실행 비용은 여러 웹 서비스 QoS 속성들의 결과값들을 종합적으로 평가한 0~1 사이의 정규값으로 가정한다.

그리고 잔존 패널티 비용은 프로세스의 종류에 따라 다르게 부여될 수 있는데, 해당 리소스를 생성했을 때 발생하는 패널티 요소가 굉장히 커서 이를 피하고 싶은 경우에는 서비스 실행 비용과의 격차를 크게 조정해서 다른 웹 서비스를 여러 개 선택하더라도 해당 리소스는 배제시킬 수 있다. 마찬가지로 리소스의 잔존 패널티가 차지하는 비중이 미미한 프로세스의 경우는 생성되는 잔존 리소스 수에 구애받지 않고 실행 비용이 저렴한 웹 서비스만을 선택하는 것이 효율적이다. 본 실험에서는 리소스의 잔존 패널티 비용이 하나의 웹 서비스가 가능한 최대 실행 비용과의 상대적 비율을 반영하기 위하여 0~1 사이의 값으로 가정하였다.

(3) 인스턴스 설정

주어진 문제에서 웹 서비스 선택은 협업 프로세스를 구성하기 위한 현재 사용가능한 리소스 정보와 필요한 리소스 정보가 주어졌을 때 두 리소스 사이를 연결하게 만드는 웹 서비스를 선택하는 것이다. 따라서

웹 서비스 네트워크에서 리소스 노드에서 가용 리소스와 목표 리소스를 새로이 정의할 때마다 새로운 인스턴스가 생성된다고 볼 수 있다.

특정 크기와 복잡도를 가진 네트워크를 생성했을 때, 해당 네트워크 구조하에서 가용 리소스와 목표 리소스에 20번의 변화를 준 20개의 인스턴스를 만들어 실험하였다. 그리고 각 인스턴스를 입력 요소로 세 개의 알고리즘을 수행하였다.

(4) 실험 환경

Branch and Bound 알고리즘의 수행은 IBM iLog의 CPLEX12.1.0을 이용하여 이루어졌으며, 네트워크 데이터 생성, 후방 검색 알고리즘 및 유방향 최소신장트리 알고리즘은 Java를 통해 구현하였다. 실험을 위해 3.2GHz AMD Phenom(tm)Ⅱ X3 710 프로세서, 4기가 바이트 RAM의 PC를 사용하였다.

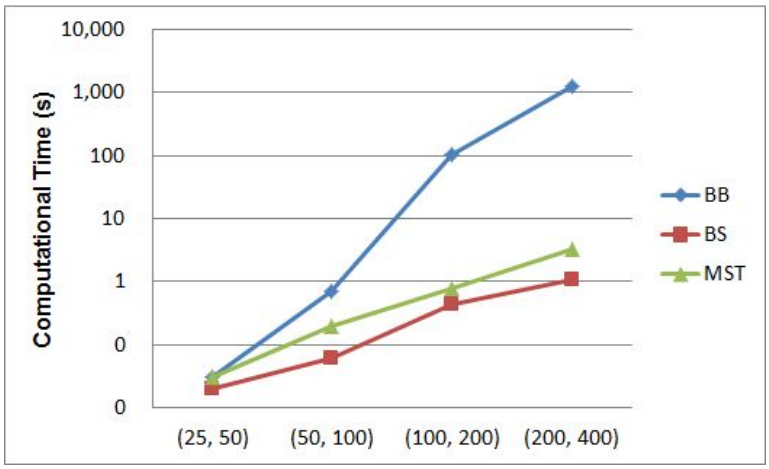
5.2. 수행 시간 비교

Branch and Bound 알고리즘은 BB, 후방 검색 알고리즘은 BS, 유방향 최소신장트리를 이용한 알고리즘은 MST로 표기하였다.

(1) 네트워크 크기에 따른 알고리즘 성능 비교

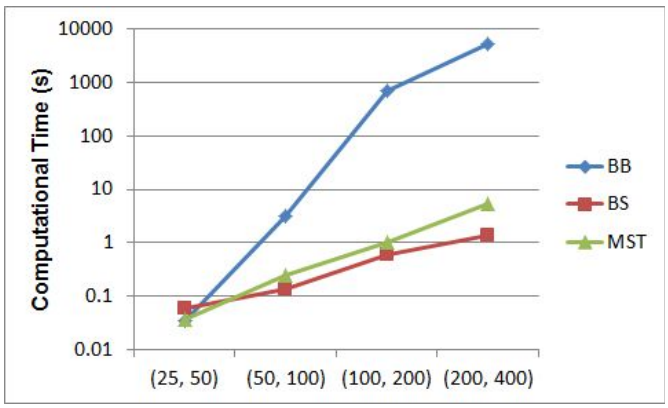
실험에서 다루는 웹 서비스 네트워크에 표현되는 웹 서비스와 리소스의 수는 (25, 50), (50, 100), (100, 200), (200, 400)로 총 네 가지의 경우를 고려하였다. 그리고 각 경우에 대한 알고리즘들의 수행 시간을 비교하였다. 수행 시간 비교를 위하여 주어진 네트워크에서 수행된 20회의 수행 시간의 평균값을 이용하였다.

- 입출력 리소스의 평균 연결수가 3인 경우



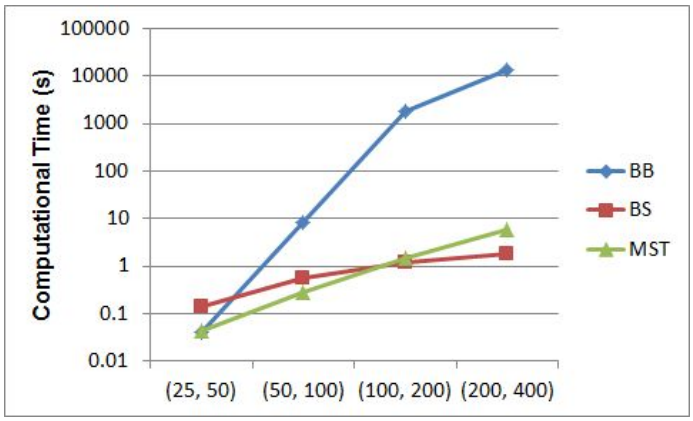
<그림 5.1> 네트워크 크기에 따른 수행 시간 1

- 입출력 리소스의 평균 연결수가 6인 경우



<그림 5.2> 네트워크 크기에 따른 수행 시간 2

- 입출력 리소스의 평균 연결수가 9인 경우



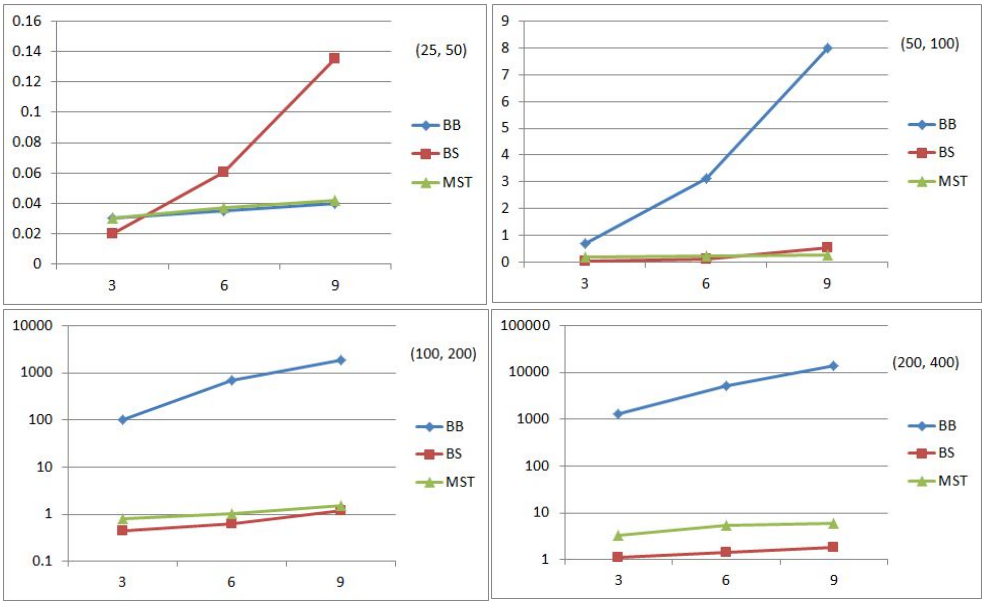
<그림 5.3> 네트워크 크기에 따른 수행 시간 3

입출력 리소스의 평균 연결수를 일정하게 유지했을 때, 각 알고리즘의 네트워크 크기에 따른 수행시간은 위 <그림 5.1, 5.2, 5.3>과 같다. Branch and Bound 알고리즘은 네트워크 크기가 커짐에 따라 수행 시간이 급속도로 증가함을 확인할 수 있으며, 수행 시간의 측면에서 봤을 때의 성능은 BB<MST<BS 순서임을 확인할 수 있다.

(2) 네트워크 밀집도에 따른 알고리즘 성능 비교

네트워크의 밀집도가 높은 경우에 따른 분석, 네트워크의 밀집도가 높다는 말은 웹 서비스들이 실행되기 위해서 많은 리소스를 필요로 하고, 실행 후에는 많은 리소스를 생성한다. 웹 서비스 노드들의 평균 연결수를 변화시켜가면서 네트워크를 생성하고, 그 때의 성능을 비교하였다. 실험에서 다루는 웹 서비스 네트워크는 입출력 리소스의 평균 연결수가 3(2~4)인 경우, 6(5~7)인 경우, 9(8~10)인 경우로 나누어 고려하였다.

수행 시간 비교를 위하여 주어진 네트워크에서 수행된 20회의 수행 시간의 평균값을 이용하였다.



<그림 5.4> 네트워크 복잡도에 따른 수행 시간

네트워크 크기를 일정하게 유지했을 때, 각 알고리즘의 네트워크 복잡도에 따른 수행시간은 위 <그림 5.4>와 같다. <그림 5.4>에서 후방 검색 알고리즘은 네트워크 크기가 작을 때는 복잡도의 영향을 받는 모습을

보였지만, 네 종류의 네트워크 크기에서 제시한 두 개의 휴리스틱 알고리즘은 복잡도의 영향이 선형적임을 알 수 있다.

5.3. 비용 비교

비용의 비교에서는 제안하는 휴리스틱 알고리즘에서 구한 비용과 최적 비용을 보장하는 Branch and Bound에서의 비용차를 확인한다. 네트워크별로 20회씩 수행한 각각의 데이터에서 오차율을 구하고 이들의 평균값을 다음 표와 같이 정리하였다.

[표 5.1] 휴리스틱 알고리즘의 비용 오차율의 평균

복잡도	크기	오차율	
		BS	MST
3	(25, 50)	1.00%	0.28%
	(50, 100)	1.43%	0.95%
	(100, 200)	1.52%	1.73%
	(200, 400)	4.31%	1.2%
6	(25, 50)	1.21%	1.25%
	(50, 100)	1.95%	1.75%
	(100, 200)	2.65%	2.15%
	(200, 400)	5.43%	2.5%
9	(25, 50)	1.29%	1.3%
	(50, 100)	2.79%	2.18%
	(100, 200)	3.11%	3.65%
	(200, 400)	5.36%	4.1%

[표 5.1]의 값을 보면, 두 알고리즘 모두 네트워크의 크기가 커질수록 오차가 커지는 모습을 보여주나 후방 검색이 그 정도가 더 큼을 확인할 수 있다. 후방 검색 알고리즘은 단계별 국소 최적해를 반복해서 최적해

를 구하기 때문이라고 보여진다. 따라서 네트워크의 크기가 커지면 후방 검색 알고리즘이 거치는 단계의 길이가 길어지게 되고 이로 인하여 오차의 크기가 커진 것으로 판단된다.

6. 결 론

6.1. 요약

기업 구조가 핵심 역량 중심으로 재편되고, 아웃소싱 형태의 협업이 강조됨에 따라 기업간으로 확장된 업무 프로세스를 효과적으로 관리하는 것이 BPM의 주요 기능으로 강조되고 있으며, 업무 프로세스의 범위가 기업 외부로 확장됨에 따라 유연성을 높이는 것이 협업 프로세스의 성공을 확보하는 주요 요인으로 인식되고 있다.

본 논문은 협업 프로세스의 유연성을 높이기 위한 웹 서비스 기반의 협업 프로세스를 동적 구성 방법을 다루었다. 대상이 되는 협업 프로세스는 워크플로우 프로세스와 웹 서비스 프로세스로 구성된다. 워크플로우 프로세스는 기업 내에서 수행되는 업무들로 이루어진 프로세스이며, 웹 서비스 프로세스는 시스템 통합을 위한 웹 서비스들과 기업간 메시지 교환을 위한 웹 서비스들로 이루어진 서브 프로세스이다. 본 연구는 협업 프로세스의 런타임 상황에서 발생하는 유연성을 다룬다.

런타임 프로세스의 유연성을 높이기 위한 기존의 연구들은 상황에 대처할 빌트인 프로세스를 미리 구축해놓고 이를 활용하는 방식을 따르고 있다. 이러한 접근은 전문적인 지식을 가진 프로세스 디자이너에 의존적이고, 모든 런타임 발생 상황을 빌트인 프로세스로 구축하는 것 역시 현실적으로 불가능하다는 단점을 갖고 있다. 이의 개선을 위하여 본 연구에서는 인스턴스 상황에 맞춰 프로세스를 동적으로 구성하는 늦은 모델링 방안을 설명하고, 런타임 프로세스의 늦은 모델링에 사용되는 서브 프로세스로 상호운용성을 보장하는 웹 서비스 프로세스를 활용한다.

웹 서비스 프로세스를 이용한 늦은 모델링을 수행하기 위하여, 본 연구는 런타임 프로세스 동적 구성 프레임워크를 제시하였고, 여기에는 웹 서비스 기반 협업 프로세스 모델에서 프로세스 파트너 독립성을 보장하는 방법과 인스턴스 정보와 웹 서비스 입출력 파라미터의 매칭을 위한 데이터 오브젝트 모델이 필요함을 설명하였다.

제시한 런타임 프로세스 동적 구성 프레임워크를 기반으로 하여 웹 서비스를 이용한 런타임 프로세스 동적 구성 메카니즘을 제안하였다. 런타임 프로세스 동적 구성에 필요한 웹 서비스를 선택하기 위해 늦은 모델링 모듈을 정의하였고, 이 모듈은 서비스 저장소에 등록된 WSDL의 웹 서비스 오퍼레이션과 파라미터 정보를 토대로 웹 서비스 네트워크 정보를 구축한다. 구축된 웹 서비스 네트워크와 현재 프로세스 인스턴스를 매칭하기 위하여 데이터 오브젝트 모델과 웹 서비스 파라미터의 유사성을 분석하는 과정이 필요하다. 매칭된 내용을 토대로 모듈은 웹 서비스 네트워크를 탐색하고 필요한 웹 서비스를 선택한다. 선택된 웹 서비스들은 WS-BPEL로 표기되어 협업 프로세스의 서브 프로세스로 작동된다.

또한 인스턴스 정보를 토대로 웹 서비스를 선택하는 과정은 네트워크 최적화 문제로 표현 가능하다. 인스턴스 정보와 연결된 웹 서비스 요소를 탐색하고 이를 선택하는 과정은 네트워크에서 더 나은 비용의 서비스 조합을 찾는 문제로 볼 수 있다. 따라서 주어진 웹 서비스 네트워크에서 비용을 최소화하는 웹 서비스 선택 최적화 문제를 정의하고, 이 문제가 NP-hard인 문제임을 증명하였다. NP-hard인 이 문제를 해결하기 위하여 유사 최적해를 제공하는 두 가지 휴리스틱, 후방 검색 알고리즘과 유방향 최소신장트리를 이용한 휴리스틱을 제안하였다.

제시한 두 개의 휴리스틱 알고리즘의 웹 서비스를 선택하는 환경에서 어느 정도의 해와 수행시간을 보이는지 확인하기 위하여 실험을 수행하였다. 실험은 웹 서비스 네트워크의 크기와 복잡도에 변화를 주고, 그 변

화의 정도에 따라 각 알고리즘들의 성능이 어떠한 모습을 보이는지 비교하였다.

6.2. 연구 의의 및 추후 연구

본 연구에서는 협업 프로세스의 유연성을 높이기 위한 웹 서비스 기반의 협업 프로세스를 동적으로 구성하는 방법을 연구 주제로 다루었다. BPM과 같은 프로세스 중심의 정보시스템이 필요성에 대한 공감대는 크지만, 널리 활용되지는 못하는 가장 큰 이유는 실시간 프로세스 변화에 대한 민첩한 지원을 못하고 있기 때문이다. 웹 서비스를 통해서 상호운용의 가능해진 환경에서 더 나은 기회 및 보충의 가능성을 열린 공간에서 찾는 노력은 협업 프로세스의 필요성과 맥을 같이 한다. 여기서는 이러한 유연성 중 런타임시에 발생하는 여러 예외 상황과 불확실성에 대비하기 위한 통로로 웹 서비스를 고려하였고, 이를 위한 프레임워크를 제안함으로써 런타임 인스턴스의 완수 및 유연한 상황 대처 가능성을 제시하였다. 본 연구의 결과물을 통해, 사용자는 런타임 인스턴스의 정보 미달의 상황을 대처할 수 있는 대체 프로세스를 얻게 되고, 대처 방안은 프로세스 디자이너에게 다시 전달되어 새로운 빌트인 프로세스 디자인의 자료로 활용될 수 있을 것으로 기대한다.

또한 런타임 프로세스 동적 구성에 참여할 웹 서비스를 탐색하고 선택하는 과정을 수리 모형화하고 이 문제의 가해성을 탐구함으로써, 실제 프로세스 시스템에 적용하기 위한 바탕을 마련했다고 볼 수 있다.

추후 연구 과제로는 웹 서비스 프로세스의 구조화가 있다. 본 논문에서 제안하는 방법으로 찾은 웹 서비스 프로세스는 직렬, 혹은 AND JOIN과 AND SPLIT으로 연결된 형태만을 허용한다. 따라서 XOR, 조건, 루프 등의 다양한 구조를 표현할 수 있는 방안을 고려할 필요가 있다. 이를 위해서는 프로세스의 시맨틱 뿐만 아니라 웹 서비스 네트워크를 표현하는 정교한 모델 역시 필요하고, 복잡한 구조를 허용하는 웹 서비스들의 QoS와도 연결되어야 한다.

또 다른 과제로는 최적화 모형의 정교화이다. 제시한 모형에서는 생성된 자원의 양을 고려하지 않고, 만들어지고 사용되었는지 여부만을 판단하였다. 그래서 웹 서비스 프로세스를 통하여 중복 생성을 하는 경우나, 한 번 생성을 하면 특정 횟수만큼의 사용을 보장하는 경우는 반영하지 못하고 있다. 따라서 자원의 양을 고려하는 방법과 특정 웹 서비스의 중복 실행을 허용함으로써 해에 포함되는 웹 서비스의 수를 줄이는 방안 역시 고려해볼 문제이다.

참고 문헌

(Aguilar-Saven, 2004) R. Aguilar-Saven, Business process modelling: Review and framework, International Journal of Production Economics, 90(2), 2004, pp.129-149.

(Akkiraju *et al.*, 2005) R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M., M.T. Schmidt, A. Sheth, and K. Verma, Web service semantics - WSDL-S, W3C Member submission, http://www.w3.org/Submission/WSDL-S/

(Alonso *et al.*, 2004) G. Alonso, F. Casati, H. Kuno, and V. Marchiraju, Web Services, Springer, 2004.

(Ardagna and Mirandola, 2010) D. Ardagna, R. Mirandola, Per-flow optimal service selection for Web services based processes, The Journal of Systems and Software, 83(8), 2010, pp.1512-1523.

(Ardagna and Pernici, 2005) D. Ardagna and B. Pernici, Global and Local QoS Constraints Guarantee in Web Service Selection, In: Proc. the IEEE International Conference on Web Services, 2005, pp.805-806.

(Bellwood *et al.*, 2003) T. Bellwood, L. Clement, and C. van Riegen, UDDI Spec Technical Committee Specification, UDDI Spec TC, 2003, http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

(Box *et al.*, 2000) D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H.F. Nielsen, S. Thatte, and D. Winer, Simple Object Access Protocol (SOAP), W3C Note, 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

(Bradley *et al.*, 1995) P. Bradley, J. Browne, S. Jackson, and H. Jagdev, Business process re-engineering(BPR) - A study of the software tools currently available, Computers and Industry, 25, 1995, pp.309-330.

(Canfora *et al.*, 2005) G. Canfora, M. D. Penta, R. Esposito and M. L.

Villani, An approach for QoS-aware service composition based on genetic algorithms, In: Proc. the 2005 conference on Genetic and evolutionary computation, 2005, pp.1069-1075.

(Cardoso *et al.*, 2004) J. Cardoso, A. Sheth, J. Miller, J. Arnold and K. Kochut, Quality of service for workflows and web service processes, Web Semantics: Science, Services and Agents on the World Wide Web, 1(3), 2004, pp.281-308.

(Carlsen *et al.*, 1997) S. Carlsen, J. Krogstie, A. Slvberg, and O.I. Lindland, Evaluating flexible workflow systems. In: Proc. the 30th Hawaii International Conference on System Sciences, 1997, pp.230-239.

(Chi *et al.*, 2008) Y. Chi, H. Lee, A formal modeling platform for composing web services, Expert Systems with Applications, 34(2), 2008, pp.1500-1507.

(Chinnici *et al.*, 2007) R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana, Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Recommendation, http://www.w3.org/TR/wsdl20/

(Christensen, 2001) E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Web Services Description Language (WSDL) 1.1, W3C Note, 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315/

(Dumas *et al.*, 2005) M. Dumas, A.H.M. ter Hofstede, and W.M.P. van der Aalst, Process Aware Information Systems, Wiley, Hoboken, 2005.

(Fellbaum, 1998) C. Fellbaum, An Electronic Lexical Database, MIT, Cambridge, 1998.

(Gulledge, 2007) T.R. Gulledge, Architecture-driven enterprise integration, International Journal of Management and Enterprise Development, 4(5), 2007, pp.535-551.

(Heinl *et al.*, 1999) P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, and M. Teschke, A comprehensive approach to flexibility in workflow

management systems, In: Proc. International joint conference on Work activities coordination and collaboration, 1999, pp.79-88.

(Jaeger *et al.* 2004) M.C. Jaeger, G.R. Goldmann and G. Muhl, QoS Aggregation for Web Service Composition using Workflow Patterns, In: Proc. 8th International Enterprise Distributed Object Computing Conference, 2004, pp.149-159.

(Kloppmann *et al.*, 2008) M. Kloppmann, D. König, F. Leymann, G. Pfau, D. Roller, Business process choreography in WebSphere. combining the power of BPEL and J2EE, IBM Systems Journal, 43(2), 2008, pp.270-296.

(Kil *et al.*, 2009) H. Kil, S. Oh, E. Elmacioglu, W. Nam, and D. Lee, Graph Theoretic Topological Analysis of Web Service Networks, World Wide Web, 12(3), 2009, pp.321-343.

(Kumar and Narasipuram, 2006) K. Kumar and M. M. Narasipuram, Defining requirements for business process flexibility, In: Proc. The 18th International Conference on Adavanced Information Systems Engineering, 2006, pp.137-148.

(Lam *et al.*, 2009) C.Y. Lam, W.H. Ip, C.W. Lau, A business process activity model and performance measurement using a time series ARIMA intervention analysis, Expert Systems with Applications, 36(3), 2009, pp.6986-6994.

(Lei and Duan, 2007) L. Lei and Z. Duan, Automating Web Service Composition for Collaborative Business Processes, In: Proc. 11th International Conference on Computer Supported Cooperative Work in Design, 2007, pp.894-899.

(Min *et al.*, 2010) Y. Min, D. Kim, T. Lim, S. Kang, A Clinical Practice Guided Decision Support System Based on Medical Ontology, International Journal of Innovative Computing, Information and Control, 6(5), 2010, pp.2361-2369.

(Mutschler *et al.*, 2008a) B. Mutschler, M. Reichert, and B. Weber,

Unleashing the effectiveness of process-oriented information systems: Problem analysis, critical success factors and implications, IEEE Transactions on Systems, Man, Cybernetics Part C, 38(3), 2008, pp.280-291.

(Mutschler *et al.* 2008b) B. Mutschler, B. Weber, and M. Reichert, Workflow management versus case handling - results form a controlled software experiment, In: Proc. 23rd Annual ACM Symposium on Applied Computing, 2008, pp.82-89.

(Oh, 2011) J. Oh, Dynamic Execution Planning for Reliable Collaborative Business Processes, Ph.D Thesis in Seoul National University, 2011.

(Oh *et al.*, 2011) J. Oh, N.W. Cho, H. Kim, Y. Min, S. Kang, Dynamic execution planning for reliable collaborative business processes, Information Sciences, 181(2), 2011, pp.351-361.

(Peltz, 2003) C. Peltz, Web Services Orchestration and Choreography, Computer, 36(10) (2003) 46-52.

(Regev *et al.*, 2007) G. Regev, I. Bider, and A. Wegmann, Defining business process lexibility with the help of invariants, Software Process Improvement and Practice, 12, 2007, pp.65-79.

(Regev *et al.*, 2006) G. Regev, P. Soffer, and R. Schmidt, Taxonomy of flexibility in business processes, In: Proc. the 7th Workshop on Business Process Modelling, Development and Support, 2006, pp.90-93.

(Regev and Wegmann, 2005) G. Regev and A. Wegmann, A regulation-based view on business process and supporting system flexibility, In: Proc. The 17th International Conference on Adavanced Information Systems Engineering, 2005, pp.91-98.

(Reichert *et al.*, 1998) M. Reichert, P. Dadam, ADEPTflex - supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems, 10(2), 1998, pp.93-129.

(Reichert *et al.*, 2005) M. Reichert, S. Rinderle, U. Kreher, P. Dadam,

Adaptive process management with ADEPT2, In: Proc. 21st International Conference on Data Engineering, 2005, pp.1113-1114.

(Reijers 2006) H.A. Reijers, Workflow flexibility: The forlorn promise, In 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), 2006, pp.271 - 272.

(Schonenberg *et al.*, 2008) H. Schonenberg, B. Weber, B. van Dongen, and W.M.P. van der Aalst, Process flexibility: A survey of contemporary approaches, In: CIAO!/EOMAS, 2008, pp.16-30.

(Shen *et al.*, 2007) W. Shen, Q. Hao, S. Wang, Y. Li, H. Ghenniwa, An agent-based service-oriented integration architecture for collaborative intelligent manufacturing, Robotics and Computer-Integrated Manufacturing, 23, 2007, pp. 315-325.

(Singh and Huhns, 2005) M.P. Singh, M.N. Huhns, Service-Oriented Computing: Semantics, Processes, Agents, John Wiley & Sons, Ltd, West Sussex, England, 2005.

(Sinur and Bell, 2003) J. Sinur and T. Bell, A BPM Taxonomy: Creating clarity in a confusing market, Gartner Research Note, T-18-9669, Gartner, Inc., Stamford, CT.

(Smith and Fingar 2003) H. Smith and P. Fingar, Business Process Management: The Third Wave, Meghan-Kiffer, Tampa, 2003.

(Snowdon *et al.*, 2007) R.A. Snowdon, B.C. Warboys, R.M. Greenwood, C.P. Holland, P.J. Kawalek, and D.R. Shaw, On the architecture and form of flexible process support, Software Process Improvement and Practice, 12, 2007, pp.21-34.

(Soffer, 2005) P. Soffer, On the notion of flexibility in business processes. In Workshop on Business Process Modeling, In: Proc. The 17th International Conference on Adavanced Information Systems Engineering, 2005, pp.35-42.

(Sun *et al.*, 2007) Y. Sun, S. He and J. Y. Leu, Syndicating Web Services: A QoS and user-driven approach, Decision Support Systems, 43(1), 2007, pp.243-255.

(van der Aalst and Hofstede, 2005) W.M.P. van der Aalst, and A.H.M. ter Hofstede, YAWL: Yet another workflow language, Information Systems, 30(4), 2005, pp.245-275.

(van der Aalst and Jablonski, 2000) W.M.P. van der Aalst and S. Jablonski, Dealing with workflow change: Identification of issues and solutions. International Journal of Computer Systems, Science, and Engineering, 15(5), 2000, pp.267-276.

(van der Aalst *et al.*, 2005) W.M.P. van der Aalst, M. Weske, D. Grünbauer, Case handling: A new paradigm for business process support, Data & Knowledge Engineering, 53(2), 2005, pp.129-162.

(Vazirani, 2000), V.V. Vazirani, Approximation Algorithm, Springer-Verlag, Berlin, 2001.

(W3C, 2004) W3C RDF Working Group, Resource Description Framework, http://www.w3.org/RDF/

(Wang *et al.*, 2006) P. Wang, K. Chao, C. Lo, C. Huang and Y. Li, A Fuzzy Model for Selection of QoS-Aware Web Services, In: Proc. the IEEE International Confernece on e-Business Engineering, 2006, pp.585-593

(Weber *et al.*, 2009) B. Weber, S. Sadiq, M. Reichert, Beyond rigidity - dynamic process lifecycle support, Computer Science - Research and Development, 23(2), 2009, pp.47-65.

(Weske, 2007) M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

(Yu *et al.*, 2007) T. Yu, Y. Zhang, and K. Lin, Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints, ACM Transactions

on the Web, 1(1), 2007, pp.6-es.

(Yuan *et al.*, 2009) Y. Yuan, X. Li and X. Zhu, Deadline Division-Based Heuristic for Cost Optimization in Workflow Scheduling, Information Sciences, 179(15), 2009, pp.2562-2575.

(Zeng *et al.*, 2004) L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering, 30(5), 2004, pp.311-328.

Abstract

Dynamic Composition for Web Service Based Collaborative Business Processes

Yeongbin Min
Department of Industrial Engineering
The Graduate School
Seoul National University

Business Process Management (BPM) technologies have been recognized as a business information technologies enabling heterogeneous systems to act as a platform for providing integration. As the scope of business process has been expanded to the outside of an organization, BPM has grown in importance to integrate disparate processes. Especially, achievement of process flexibility is a key success factor to manage collaborative business processes.

The runtime flexibility of a collaborative process is an important factor for most practical BPM systems. As a collaborative process is composed by independently developed component web services, fast and appropriate selection is a key factor for success. Recently, many researchers have proposed web service selection methodologies for efficient service composition. Most of them assume that composition topology, i.e. logical flow is fixed. However, fixed composition

topology limits not only excessive dependence on domain experts, who design abstract process flow, but also impossibility of considerations of all situations that may occur.

This dissertation focuses on dynamic composition of collaborative processes in web service environments. To overcome the limits of previous researches, late modeling method, which enables dynamic composition of process in accordance with the process instances is proposed. In detail, the framework of executing an incomplete process instance at runtime, i.e., one which does not contain sufficient information to allow it to be executed to completion is considered.

In this research, first, an integrated runtime process composition framework and its mechanism are suggested. In this process flexibility escalation framework, the problem of instance underspecification is dealt with use of web service network and selection of partial network, which is satisfying instance conditions. Since determination of collaborative process, which is part of web service network is classified as a NP-hard problem, two heuristic algorithms are developed. Additionally, to ensure the efficiencies of proposed heuristic algorithms, a set of experiments of comparing the proposed algorithms' performances with those of the branch-and-bound method are conducted.

Keywords: BPM, process flexibility, runtime process composition, web service composition
Student number: 2007-30166