효율적인 서비스 모니터링을 위한 모니터링 가능한 서비스 설계 기법

이현민O 김수동

숭실대학교

hmlee@otlab.ssu.ac.kr, sdkim777@gmail.com

A Method to Designing Managed Service for Efficient Service Monitoring

Hyun Min LeeO Soo Dong Kim

Soong-Sil University

요 약

서비스 지향 컴퓨팅은 서비스를 동적으로 조합하여 사용자가 요구하는 비즈니스 프로세스를 개발하는 효율적인 개발 방법이다. 서비스는 분산환경에 배치되고 인터페이스만을 외부로 노출하여 내부의 세부사항은 숨긴다. 이러한 서비스의 특징은 서비스 사용자가 서비스의 품질을 알기 어렵게 만든다. 서비스 소비자가 중요한 도메인에서 서비스를 사용하려면 서비스 품질(QoS)이 보장되어야 한다. 서비스 품질을 보장하기 위해서는 서비스 품질 측정이 먼저 선행되어야 한다. 하지만 현재까지 서비스 품질 모델의 표준화가 이루어 지지 않아 모니터링 해야 할 명확한 대상이 정의되지 않았고, 그로 인해 서비스 모니터링 방법 및 모니터링 효율성에 대한 연구는 거의 이루어 지지 않았다. 이와 같은 문제를 해결하기 위하여, 본 논문에서는 먼저 서비스 모니터링 해야 할 데이터를 명확하게 정의한다. 또한 정의된 모니터링 데이터를 분류하고 분류된 데이터의 수집을 위한 모니터링 가능한 서비스 설계방법을 제안한다. 마지막으로, 실험을 통해 제안된 기법의 실효성과 효율성을 보여준다.

1. 서 론

서비스 지향 컴퓨팅은 재사용성이 높은 서비스를 이용하여 어플리케이션을 개발하는 효율적인 개발 방법이다 [1]. 서비스 개발자는 UDDI(Universal Description Discovery and Integration)을 이용하여 서비스를 검색하고 BPEL(Business Process Execution Language)을 사용하여 서비스를 조합한다.

서비스 소비자는 서비스 인터페이스를 통하여 서비스의 기능성을 사용한다. 따라서 서비스는 서비스 소비자에게 블랙박스 형태로 인식된다. 서비스의 이런 특징은 서비스 소비자가 서비스의 품질을 알기 어렵게 만들고 서비스 제공자가 배포한 서비스에 문제가 발생하여도 서비스 소비자가 서비스의 문제를 직접적으로 해결할 수 없게 만든다.

서비스 소비자가 중요한 도메인에서 서비스를 사용하려면 서비스의 품질이 보장되어야 한다. 즉, 서비스의 품질이 정확히 측정되어야 하고 품질이 떨어졌을 경우 그에 따른 조치가 이루어 져야 한다. 서비스 모니터링은 서비스, 프로세스, 관련된 환경에서 모니터링 정보를 수집하는 활동이다 [2]. 서비스의 품질을 측정하기 위해서는 서비스 모니터링을 통해 단일 서비스 및 복합서비스, 비즈니스 프로세스, ESB(Enterprise Service Bus)등에서 품질 측정에 필요한 모든 데이터를 수집해야 한다. 하지만 현재까지 서비스 품질 모델의 표준화가 이루어 지지 않아 모니터링 해야 할 데이터가 명확하지 않다 [3].

그 결과 지금까지 서비스 모니터링을 위한 연구는 미들웨어를 이용하여 모니터링 데이터를 획득하는 방법을 중심으로 연구되었다. 이는 수집 할 수 있는 데이터가 한정되어 있어 품질을 측정하기 위한 데이터를 수집하는 데에 한계가 있다. 또한 미들웨어에서 제공하는 데이터를 중복으로 가져오게 되어 모니터링으로 인한 오버헤드 문제가 발생하게 된다. 여기서 오버헤드란 모니터링으로 인해 사용되는 추가적으로 사용되는 자원 및 시간을 의미한다. 그러므로 모니터링 해야 할 데이터를 명확히 정의하고 그 데이터를 효율적으로 모니터링 하는 방법이 필요하다.

이와 같은 문제를 해결하기 위하여 본 논문에서는 모니터링 가능한 서비스 설계 기법을 제안한다. 먼저 3장에서는 모니터링 해야 할 대상을 명확하게 정의한다. 4장에서는 정의된 데이터를 분류하고 이를 수집 할 수 있는 기법을 제안한다. 또한 모니터링 효율성을 측정하는 방법을 제시한다. 5장에서는 제안된 모니터링 기법을 지원하는 모니터링 가능한 서비스 설계방법을 제안한다. 6장에서는 실험을 실시하여 제안된 방법의 실효성과 효율성을 보여준다.

2. 관련 연구

서비스 모니터링은 QoS를 구하기 위해 데이터를 획득하는 것부터 측정된 QoS를 필요로 하는 관리자 혹은 소비자에게 보여주는 것까지의 과정을 모두 포함한다 [2]. 모니터링 데이터를 획득하는 측면에서 보면, Zeng et al. [4]은 QoS를 구하기 위한 품질 메트릭을 3가지로 구분하였다. 이는 서비스 제공자가 제공하는 메트릭, 소비자가 평가하는 메트릭, 그리고 관찰 가능한 메트릭이다. 이 중 세 번째 메트릭을 모니터링 대상으로 정의하고 이를 구하기 위한 서비스 모니터링 아키텍처를 제안하였다. 하지만 모니터링 데이터를 가져오는 과정이 어렵지 않음을 가정하여 획득한 데이터를 가공하여 메트릭을 계산하는 과정을 중점적으로 다룸으로써 모니터링 데이터를 획득하는 과정에 대한 설명이 부족하다. 또한 Artaiam et al. [5]은 기존의 미들웨어를 확장한 QoS 모니터링 모델을 제안하였다. 먼저 QoS 모델을 6가지의 속성으로 정의하고 이를 AMX(Application Server Management Extension)을 이용하여 획득 가능하도록 Java System Application Server를 확장하였다. 이는 특정 플랫폼 안에서만 동작하도록 설계되어 있어 다양한 환경에서 설계된 모든 서비스를 모니터링 하기 어렵다. Baresi et al. [6]는 BPEL 엔진을 활용한 서비스 모니터링 기법을 제안하였다. Dynamo는 관점지향 프로그래밍(Aspect-Oriented Programming, AOP)을 사용하여 BPEL 프로세스가 동작하는 동안 BEPL 엔진이 수집할 수 있는 모니터링 데이터를 얻어 오는 방법이다. Astro는 독립된 소프트웨어 모듈을 사용하여 RTML(run-time monitor specification Language)로 정의된 속성을 확인하는 방법이다. 하지만 BPEL에 별도의 모니터링을 위한 작업이 추가되어야 하는 것에 비해 미들웨어에서 획득할 수 있는 모니터링 데이터와 큰 차이가 없어 실효성 면에서 한계가 있다.

3. 서비스 품질 모델

OASIS(Organization for the Advancement of Structured Information Standards)는 웹서비스를 위한 품질 모델(WSQM)을 발표하였다 [7]. WSQM은 품질 요소, 품질 활동, 품질 관계자의 세가지로 구성되어 있고 이중 품질 요소에 관하여 살펴보면 다음과 같다.

첫째로 비즈니스 가치 품질이 있다. 이는 웹서비스의 비즈니스 측면을 평가한다. 예를 들면 서비스의 가격, 비즈니스 적합성, 비즈니스 효과, 인식, 명성 등이 있다. 둘째로 서비스 단계의 측정 품질이 있다. 이는 웹서비스가 사용되는 동안 측정되는 정량적인 속성이다. 예를 들면 응답시간, 최대 처리량, 이용가능성, 접근성 등이 있다. 또한 표준 준수성, 비즈니스 프로세스 품질, 관리 품질, 보안 품질 등으로 구성된다.

하지만 WSQM이 위와 같은 모델을 제공함에도 불구하고 서비스 품질과 관련된 다른 논문을 살펴보면 WSQM을 사용하지 않는다. 이는 아직까지 폭넓게 인정받고 있는 서비스 품질 모델의 표준이 아니기 때문이다. 따라서 우리는 서비스에 특화되어 있지는 않지만 표준으로 인정 받고 있는 ISO9126을 사용하여 모니터링 해야 할 대상을 정의한다.

그림 1과 같이 서비스 품질 모델에는 여러 개의 품질 속성이 있고, 품질 속성에는 여러 개의 품질 메트릭으로 구성되며, 품질 메트릭은 메트릭을 구하기 위한 메트릭 매개변수로 이루어져 있다 [8]. 품질 메트릭을 구하기 위한 메트릭 매개변수가 바로 서비스 모니터링을 통해 수집해야 할 데이터가 된다.

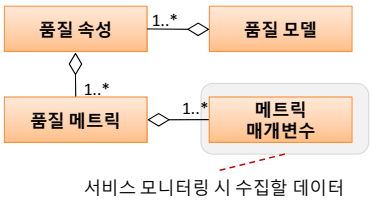

그림 1 서비스 품질 모델의 메타모델

지금까지 제안된 품질 모델과 ISO9126을 종합하여 우리는 그림 2와 같은 네 가지를 모니터링 해야 할 대표적인 품질 속성으로 선택한다.

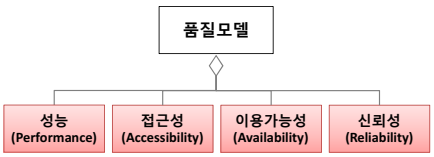

그림 2 서비스 품질 모델

성능, 접근성, 이용가능성은 거의 모든 QoS에 들어가 있을 만큼 보편적인 속성이고 신뢰성은 중요한 도메인에서 서비스를 사용할 때 가장 중요한 속성이라 품질 모델에 포함된다.

성능은 서비스 응답시간과 처리량을 구하기 위한 메트릭으로 구성되어 있고 응답시간의 대표적인 메트릭은 아래와 같다.

$$SOPT = \begin{pmatrix} ServiceOperationEndTime - \\ ServiceOperationStartTime \end{pmatrix}$$

*SOPT(Service Operation Time)*은 서비스가 호출되어 운영된 시간을 측정하는 메트릭으로 *ServiceOperationStartTime*은 서비스가 실제 수행되기 시작한 시간이고 *ServiceOperationEndTime*은 서비스가 수행을 마친 시간이다.

또한 신뢰성은 회복성과 결함허용성을 구하기 위한 메트릭으로 구성되어 있고 결함허용성의 대표적인 메트릭은 아래와 같다.

$$FATO = \left(\frac{NumberofHandledException}{NumberofServiceException} \right)$$

*FATO(Fault Tolerance)*는 서비스에 예외상황이 발생하였을 때 대처할 수 있는 정도를 측정하는 메트릭으로 *NumberofHandledException*은 서비스에서 문제없이 처리한 예외상황의 개수를 나타내고 *NumberofServiceException*은 서비스에서 발생한 예외상황의 개수를 나타낸다.

4. 모니터링 데이터 및 기법 분류와 모니터링 효율성

이 장에서는 모니터링 수집하는 시간에 따라 데이터를 분류하고 데이터를 수집하는 기법을 분류한다. 또한 모니터링 효율성을 측정하는 방법을 제시한다.

4.1 모니터링 데이터의 분류와 데이터 수집 기법

모니터링 데이터는 다음과 같이 분류 할 수 있다.

- 정기적인 데이터와 비정기적인 데이터

먼저 모니터링 데이터는 수집하는 시간을 기준으로 정기적인 데이터와 비정기적인 데이터로 분류된다. 정기적인 데이터는 일정한 간격으로 수집되는 데이터이다. 예를 들면 일정 시간 마다 체크해야 하는 상태 데이터는 정기적인 데이터라고 할 수 있다. 비정기적인 데이터는 일정한 간격이나 규칙 없이 수집되는 데이터이다. 비정기적인 데이터는 다시 세 가지로 분류된다. 즉, 모든 상태 변화 때마다 수집되는 모든 상태 변화 데이터와 이벤트 발생시에 수집되는 이벤트 발생 데이터, 그리고 모니터 에이전트의 요청 시마다 수집되는 요청 데이터가 있다. 여기서 모니터 에이전트는 서비스 혹은 미들웨어를 통해 모니터링 데이터를 수집하는 모듈이다.

모니터링 데이터를 수집하는 기법은 다음과 같이 분류 할 수 있다.

- 모니터링 환경을 이용하여 수집하는 기법과 서비스의 모니터링 기능을 이용하여 수집하는 기법

모니터링 데이터를 수집하는 기법은 모니터링 환경을 이용하여 수집하는 기법과 서비스의 모니터링 기능을 이용하여 수집하는 기법으로 분류된다. 서비스 모니터링 기능 이용하여 수집하는 기법은 다시 서비스의 모니터링 인터페이스를 이용하여 수집하는 기법과 서비스가 모니터 에이전트에 전송하여 수집하는 기법으로 분류된다.

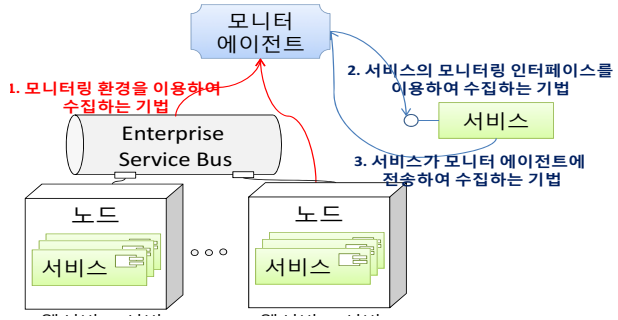

그림 3 모니터링 데이터를 수집하는 기법

그림 3은 앞에서 설명한 세 가지의 모니터링 데이터를 수집하는 기법을 보여주고 있다. 모니터링 환경을 이용하여 수집하는 방법은 모니터 에이전트가 웹서비스 서버나 ESB등의 미들웨어를 활용하여 모니터링 데이터를 수집하는 것이다. 서비스의 모니터링 기능을 이용하여 수집하는 방법은 서비스에 붙어 있는 모니터링 인터페이스를 이용하여 수집하거나 서비스가 직접 모니터 에이전트에 데이터를 전송하는 기능을 사용하여 데이터를 수집하는 것이다. 본 논문에서는 전자를 수동적인 모니터링, 후자를 능동적인 모니터링 기법이라고 한다.

그림 4는 수집하는 시간에 따라 분류된 데이터와 모니터링 데이터를 수집하는 기법에 매핑 시킨 그림이다. 화살표가 의미하는 것은 다음과 같다. 예를 들면 정기적인 데이터는 모니터링 환경을 이용하여 수집할 수 있을 뿐만 아니라 서비스 인터페이스를 이용하거나 서비스의 전송 기능을 이용하여 수집 가능하다.

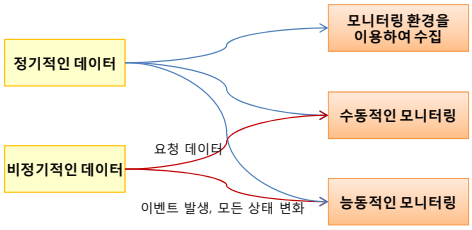

그림 4 분류된 데이터와 데이터 수집기법의 매핑관계

4.2 모니터링 효율성

본 논문에서는 모니터링 효율성을 '모니터링을 수행하면서 데이터 수집량을 최소화 하는 것'이라고 정의한다. 모니터링의 효율성을 측정하기 위해서는 서비스에서 모니터링 에이전트로 전송되는 데이터의 양을 계산 할 필요가 있다. 다음은 수집하는 시간에 따른 모니터링 데이터의 양을 계산하는 식이다. 모든 데이터의 1회 전송량은 *a*로 같다라고 가정하면, 모니터링 데이터의 i번째 전송량은 a_i라고 나타낼 수 있다. 그리고 *TotalTransfers*를 전송량의 합이라고 하고, 전체시간을 *t*라고 한다.

정기적인 데이터가 *t*동안에 전송된 전송량은

$$TotalTransfers = \sum_{i}^{n} a_i$$ 라고 할 수 있다. 전송된 횟수 *n*은 *t>interval*(전송간격)이라고 할 때 *t/interval*의 몫이 된다. 이는 정확한 전송량 예측이 가능하다.

또한 *t*동안에 일어난 상태변화 횟수를 *c*, 그리고 상태변화에 따라 발생한 이벤트 횟수를 *e*라고 하면 모든 상태 변화 데이터의 전송량과 이벤트 데이터의 전송량은 각각,

$$TotalTransfers = \sum_{i}^{c} a_i \,,\quad TotalTransfers = \sum_{i}^{e} a_i$$ 라고

할 수 있다. 두 데이터의 전송량은 통계자료에 의해 어느 정도의 예측이 가능하다.

그리고 요청 데이터의 전송량을 계산하면 다음과 같다. *t*동안에 모니터링 데이터 요청 횟수를 *r*이라 하면 필요한 데이터의 데이터 전송량은

$$TotalTransfers = \sum_{i}^{r} a_i$$ 라고 할 수 있다. 이 데이터의 전송량은 예측이 불가능하다.

여기서 같은 시간 *t*동안 전송된 전송량은 전송된 횟수에 비례하기 때문에 전송된 횟수를 비교하면 각 데이터의 전송량을 알아볼 수 있다. 먼저 모든 상태 변화에 대해 모니터링 데이터를 전송하는 경우 전송량은 다른 어떤 데이터의 전송량 보다 많다, 즉, *c>n, e, r* 이다. 모든 상태 변화가 이벤트의 발생을 의미하지 않기 때문에 *c*는 *e*보다 크다. 또한 *n>c* 라면 중복되는 데이터가 전송됨을 의미하기 때문에 정기적인 데이터를 생성하는 의미가 없다. 요청횟수 *r* 또한 *c*보다 클 경우 모든 상태 변화를 선택하는 게 효율적이기 때문에 *c>r*이 된다. 모니터링 데이터를 상태 변화 데이터로 분류하면 네 가지 중 전송량이 가장 크기 때문에 모니터링 데이터 분류 시 그 점을 고려하여 결정해야 한다.

n, e, r 은 절대적인 비교가 불가능하다. 따라서 상황에 따라 데이터를 분류해야 한다. 데이터를 누적시킬 필요가 없고 현재 서비스의 상태 확인을 위한 데이터는 요청 데이터로 분류하는 것이 좋다. 요청 데이터는 수동적인 모니터링 기법으로 구현이 가능하다. 이는 서비스에 최소한의 추가 자원만을 요구하기 때문에 모니터링 오버헤드가 적게 발생한다. 모든 상태 변화를 전송할 필요는 없지만 서비스의 품질이 특정 이벤트에 종속적인 경우 이벤트 발생 데이터로 분류한다. 이는 서비스가 이벤트 발생을 감지하여 모니터 에이전트에 알려야 하기 때문에 서비스에 추가로 많은 자원이 소모된다. 정기적인 데이터는 일반적으로 통계를 내기 위해 일정시간 마다 누적되는 데이터를 의미한다. 정기적인 데이터로의 분류 시 우선적으로 고려해야 할 사항은 데이터를 가져오는 시간간격이다. *c*와 *e*사이의 데이터를 전송하도록 설정하는 것이 데이터의 손실과 중복을 최소화 할 수 있다. *e*는 통계를 사용하여 예측 가능하기 때문에 적절한 시간간격의 설정이 가능하다.

본 논문에서는 모니터링 환경을 이용해서 모니터링 데이터를 수집하는 방법을 다루지 않는다. 그 이유는 다음과 같다. 먼저 모니터링 환경을 이용해서 수집할 수 있는 데이터는 정기적인 데이터로 제한 된다. 또한 분류된 다른 데이터를 수집하기 위해서는 모니터링 환경에 대한 수정이 필요한데 이는 특정 플랫폼마다 차이가 있어 실용성이 떨어진다.

5. 모니터링 가능한 서비스 설계

이 장에서는 앞에서 분류한 모니터링 데이터의 수집을 위해 수동적인 모니터링이 가능한 서비스와 능동적인 모니터링이 가능한 서비스를 설계한다.

5.1 수동적인 모니터링이 가능한 서비스 설계

수동적인 모니터링이란 모니터 에이전트가 모니터링의 대상인 서비스에 모니터링 데이터를 요청하여 수집하는 것이다. 모니터링 데이터를 수동적으로 수집하기 위해서 서비스는 모니터 에이전트가 접근할 수 있도록 모니터링 인터페이스를 제공해야 한다. 그림 5와 같이 수동적인 모니터링이 가능한 서비스는 서비스와 모니터링 인터페이스 그리고 모니터링 데이터 저장소로 구성되어 있다. 서비스는 상태 값을 포함하고 있다.

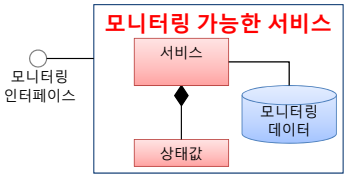

그림 5 수동적인 모니터링이 가능한 서비스의 구조

수동적인 모니터링을 위해 서비스에 필요한 요소의 기능에 대해 살펴보면 다음과 같다.

- 서비스는 모니터링 데이터 저장소에 모니터링 데이터를 저장한다. 서비스의 상태 값은 서비스의 현재 상태를 나타낸다.
- 모니터링 인터페이스는 모니터 에이전트가 모니터링 가능한 서비스에 접근할 수 있는 지점이다. 모니터 에이전트는 모니터링 인터페이스를 통하여 서비스의 현재 상태 값 또는 모니터링 데이터를 요청한다.

수동적인 모니터링 기법을 사용한 모니터 에이전트와 서비스 사이의 모니터링 데이터 전송과정을 시퀀스 다이어그램으로 나타내면 그림 6과 같다.

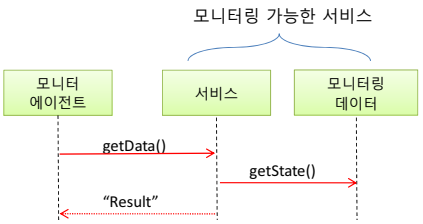

그림 6 수동적인 모니터링 시퀀스 다이어그램

모니터 에이전트는 모니터링 인터페이스를 통하여 서비스에 모니터링 데이터를 요청한다. 서비스는 모니터 에이전트가 요청한 데이터를 획득한 후 요청한 모니터링 데이터를 모니터 에이전트에 전송한다.

5.2 능동적인 모니터링이 가능한 서비스 설계

능동적인 모니터링이란 모니터링의 대상인 서비스가 모니터 에이전트에 모니터링 데이터를 전송하는 것이다.

서비스가 능동적으로 모니터 에이전트에 모니터링 데이터를 전송하기 위해서는 서비스가 데이터를 전송할 수 있는 기능을 가지고 있어야 한다. 즉, 그림 7에서와 같이 모니터링 가능한 서비스는 서비스와 모니터링 데이터를 보관하는 저장소, 에이전트 정보 관리 모듈, 스케줄러, 데이터 전송 모듈로 구성되어 있다.

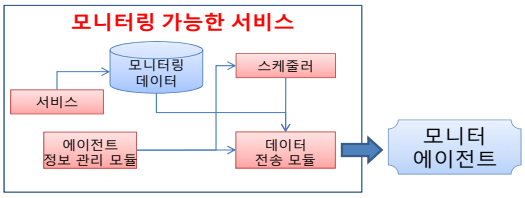

그림 7 능동적인 모니터링이 가능한 서비스의 구조

능동적인 모니터링을 위해 서비스에 필요한 요소와 기능에 대해 하나씩 살펴보면 다음과 같다.

- 서비스는 저장소에 모니터링 데이터를 저장한다.
- 에이전트 정보 관리 모듈은 서비스의 모니터링 데이터를 필요로 하는 모니터 에이전트의 주소와 원하는 모니터링 데이터, 그리고 모니터링 데이터를 받기 원하는 시간 정보를 가지고 있다. 이 중 시간 정보는 스케줄러에 알려주고 에이전트의 주소는 데이터 전송 모듈에 알려준다.
- 스케줄러는 각 데이터에 대한 전송 정책을 가지고 있다. 모니터링 데이터 별로 혹은 에이전트 별로 데이터에 대한 전송 정책이 다르기 때문에 스케줄러는 에이전트 정보 관리 모듈로부터 전송정책을 입력 받아 정책에 맞추어 모니터링 데이터가 전송될 수 있도록 데이터 전송 모듈에 전달 시기를 알린다.
- 데이터 전송 모듈은 실제로 모니터링 데이터를 각 에이전트에 전송하는 역할을 한다. 데이터 전송 모듈은 스케줄러의 알림을 받은 뒤 필요한 모니터링 데이터를 저장소에서 확보한다. 그리고 에이전트 매니저로부터 모니터 에이전트의 주소 및 현재 상태에 관한 정보를 받아 모니터링 데이터를 모니터 에이전트에 전송한다.

능동적인 모니터링 기법을 사용한 모니터 에이전트와 서비스 사이의 모니터링 데이터 전송 과정을 시퀀스 다이어그램으로 나타내면 그림 8과 같다. 모니터 에이전트는 서비스에 자신의 정보를 등록시키는 것으로 모니터링 데이터를 전송 받을 준비를 한다. 에이전트 정보 관리 모듈은 모니터링 에이전트가 등록하면 에이전트의 정보를 보관한다. 그리고 스케줄러에 모니터링 데이터 전송 정책을 전달한다. 결정된 전송 정책에 따라 스케줄러는 데이터 전송 모듈에 데이터 전송 시간을 알려준다. 데이터 전송 모듈은 에이전트 정보 관리 모듈에게서 모니터링 에이전트의 주소를 요청하고 모니터링 에이전트에 모니터링 데이터를 전송한다.

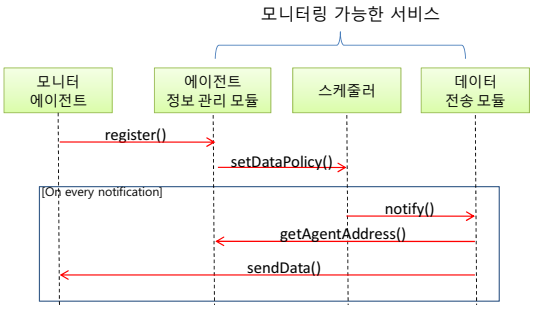

그림 8 능동적인 모니터링 시퀀스 다이어그램

능동적인 모니터링이 가능한 서비스는 요청 데이터를 제외하고 앞에서 분류된 모든 모니터링 데이터 수집이 가능하다. 모니터 에이전트는 에이전트 매니저에게 데이터 전송 정책을 전달 할 수 있기 때문에 정기적인 데이터 수집을 원하는 경우 스케줄러에 해당 기간을 설정하여 정기적으로 알림을 하여 전송한다. 서비스의 상태 변화 혹은 이벤트 발생은 서비스에서 발생을 감지할 수 있기 때문에 상태 변화나 이벤트가 발생 하면 스케줄러에서 데이터 전송 모듈에 알려서 데이터를 전송한다.

6. 실험

6.1 시나리오

이 실험의 시나리오는 다음과 같다. 같은 기능을 수행하는 시뮬레이션 서비스 4개를 같은 환경에 배치한다. 세 개의 서비스는 호출됨과 동시에 각 서비스는 한가지 종류의 데이터를 전송한다. 예를 들면 첫번째 서비스는 모든 데이터를 정기적인 데이터로 분류하여 전송한다. 마지막 서비스는 최적화된 모니터링 데이터 전송을 한다. 모니터링 데이터는 *SOFT*와 *FATO*를 구하기 위하여 *ServiceOperationStartTime, ServiceOperationEndTime, NumberofHandledException, NumberofSvcException* 그리고 호출된 서비스의 반환값을 모니터 에이전트에 전송하도록 한다.

다음은 실험 시나리오에 적용될 실험 환경이다. 본 실험에서는 시뮬레이션 웹 서비스를 *Glassfish V2* 서버에 각기 다른 설정으로 배치하였다. *Glassfish V2* 서버 컴퓨터의 사양은 *CPU* 인텔 펜티엄 *3.0GHz*, 메모리 *2GB* 이고, 운영체제로는 *Windows XP*를 사용한다.

6.2 실험 및 평가

이 실험에서는 그림 9와 같은 유저 인터페이스를 사용하였다. 동일한 실험 환경을 위해 모든 서비스는 1000번을 호출 하였고 호출 사이의 간격은 100ms로 통제 하였다. 또한 서비스 에이전트의 역할이 필요한 정기적인 데이터는 500ms의 시간 간격을 주었다.

그림 9 실험을 위한 유저 인터페이스

실험을 통해 얻은 결과는 표 1과 같다. 데이터 전송량과 중복 데이터의 양은 모니터링 데이터가 저장되어 있는 *MySQL*을 사용하여 측정하였고 서비스 평균 응답 시간은 서비스 호출 시 앞뒤 시간을 기록하여 측정하였다.

표 1 서비스 모니터링 실험 결과

	효율성 기준	측정량
서비스1 (정기적인 데이터)	데이터 전송량/중복량	218KB/78KB
	서비스 응답시간	229ms
서비스2 (모든 상태 변화)	데이터 전송량/중복량	331KB/132KB
	서비스 응답시간	242ms
서비스3 (이벤트 발생)	데이터 전송량/중복량	181KB/2KB
	서비스 응답시간	244ms
서비스4 (최적화)	데이터 전송량/중복량	85KB/1KB
	서비스 응답시간	245ms

표 1에서 서비스2와 서비스3의 데이터 전송량이 차이가 나는 이유는 데이터 *NumberofHandledException*와 *NumberofSvcException*의 경우 예외상황이 발생하지 않으면 이벤트가 발생하지 않기 때문에 이 경우 서비스 2만 데이터를 전송하기 때문이다. 서비스1과 서비스3의 데이터 전송량은 서비스1의 시간간격에 따라 달라진다. 여기서는 시간간격이 평균 이벤트 발생 간격보다 상대적으로 좁았기 때문에 서비스1의 전송량이 많았다. 변화가 거의 없는 모니터링 데이터의 경우에는 서비스1,2와 같이 중복 데이터가 많이 발생하게 되어 효율성이 많이 떨어지게 된다. 모니터링 기능이 서비스 성능에 주는 영향은 수동적인 모니터링 기법을 사용한 서비스1이 가장 적다. *SOFT*는 정기적인 데이터, *FATO*는 이벤트 발생, 그리고 반환값은 모든 상태 변화로 데이터를 분류하여 최적화 시킨 서비스4는 같은 정보를 거의 중복량 없이 다른 서비스와 비교하여 약 1/4의 전송량으로 해결함으로써 모니터링의 효율성을 보여주고 있다.

7. 결론

서비스는 서비스 소비자에게 블랙박스 형태로 인식되기 때문에 소비자는 서비스의 품질을 알기 어렵다. 중요한 도메인에서 서비스를 사용하려면 서비스의 품질이 정확히 측정되어 품질이 보장되어야 한다. 하지만 현재까지 서비스 모니터링 연구는 서비스 품질 측정을 위해 요구되는 데이터 보다는 미들웨어에서 제공하는 데이터를 중심으로 연구되어 그 한계가 있었다.

본 논문에서는 이와 같은 문제를 해결하기 위해 모니터링 해야 할 데이터를 명확히 정의하였고 정의 된 데이터 집합을 분류하고 이를 수집 할 수 있는 기법을 제안하였다. 또한 제안된 기법을 지원하는 모니터링 가능한 서비스 설계 방법을 제안하고 실험을 통해 모니터링 효율성을 보여주었다.

Acknowledgement

본 과제는 정보통신산업진흥원의 SW공학 요소기술 연구개발사업의 결과물임을 밝힙니다.

참고문헌

[1] Erl, T., *SOA Principles of Service Design*, Prentice Hall, July, 2007.

[2] Papazoglou, M., Traverso, P., Dustdar, S., and Leymann, F., “Service-Oriented Computing: State of the Art and Research Challenges,” *IEEE Computer*, Vol. 40, No. 11, pp. 38-45, Oct. 2007.

[3] Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., and H. Chang, “QoS-Aware Middleware for Web Services Composition”, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp. 311-327, May 2004.

[4] Zeng, L., Lei, H., and Chang, H., “Monitoring the QoS for Web Services,” *In Proceedings of the 5th international conference on Service-Oriented Computing,* Vol. 4749, pp. 132-144, 2007.

[5] Artaiam, N., and Senivongse, T., “Enhancing Service-Side QoS Monitoring for Web Services,” *In Proceedings of Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing,* pp. 765-770, 2008.

[6] Baresi, L., Guinea, S., Pistore, and M., Trainotti, M., “Dynamo + Astro: An Integrated Approach for BPEL Monitoring,” *In Proceedings of the IEEE International Conference on Web Services,* pp. 230-237, 2009.

[7] Kim., E. and Lee., Y., *WSQM—Quality Model for Web Services*, OASIS (To be appeared). http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc

[8] ISO/IEC, *Software Engineering—Product Quality—Part 1: Quality Model. ISO/IEC 9126-1*, June, 2001.