ISSN(Print): 2287-5905

http://dx.doi.org/10.3745/KTSDE.2012.1.3.149

An Aspect-based Testing Framework for Performance Evaluation of Composite Service

Jong-Phil Kim† · Jang-Eui Hong††

ABSTRACT

As service-oriented software is considered as one of solutions to fulfill the users' needs in internet service environment, it has been increased the demands of reliable service development by the composition of internet services. However a critical issue in the service development approach is to satisfy the performance requirement as well as the functional correctness for the developing services, because impatient user multiply clicks the request-button of service without a short waiting. This paper proposes a framework to examine the performance of composite service. Our testing framework provides the data of service response time to service developer by measuring the service execution time. We develope an Aspect-based timer service, and weave the service with existing services to measure the execution time. Additionally, we perform some experiments to confirm the usefulness of performance test for composite service. This framework can support to develop a good performance service by substitution of the dragging service with another new service that will be a component of composite service.

Keywords : Composite Service, Performance Testing, Testing Framework

조합된 서비스의 성능 평가를 위한 Aspect 기반 테스팅 프레임워크

김 종 필† · 홍 장 의††

요 약

최근 서비스 기반의 소프트웨어 개발이 사용자의 다양한 요구를 충족시킬 수 있는 하나의 솔루션으로 부각되면서, 안정적인 서비스의 조합을 통하여 보다 큰 서비스를 제공하려는 시도가 증가하고 있다. 그러나 조합된 서비스의 개발시 고려되어야 하는 사항중의 하나는 사용자의 입장에서 서비스의 정확성과 함께 신속성을 제공해야 한다는 것이다. 왜냐하면 사용자가 서비스의 요청 과정에서 늦은 응답으로 인하여 실행 버튼을 중복적으로 클릭하는 행동을 보이기 때문이다. 본 논문에서는 조합된 서비스의 성능을 측정하기 위한 테스팅 프레임워크를 제시한다. 즉, 조합된 서비스의 실행 시간을 측정함으로써, 개발자에게 서비스의 성능을 분석할 수 있는 도구를 제공한다. 이러한 실행시간 측정을 위하여 본 연구에서는 Aspect 컴포넌트를 이용하는 타이머 서비스를 개발하여 기존 웹 서비스들과 연동할 수 있도록 하였다. 또한 몇 실험을 통하여 조합된 서비스의 성능 테스트가 가능함을 확인하였다. 제시한 프레임워크는 조합된 서비스를 구성하는 단위 서비스 중에서 가장 많은 시간이 소요되는 서비스를 식별하고 필요에 따라 다른 서비스로 대체할 수 있는 서비스 개발을 가능하게 한다.

키워드 : 조합서비스, 성능 테스트, 테스팅 프레임워크

1. 서 론

최근 스마트 폰과 같은 모바일 디바이스의 보급으로 인하여 다양한 웹 서비스를 지원하기 위한 기술이 연구되고 있다. 그 중에서 대표적인 연구는 개별적인 웹 서비스를 조합하여 하나의 큰 서비스로 제공하고자 하는 것이다. 예를 들면 Fig. 1과 같이 여행 서비스를 제공하기 위하여 개별적인 항공예약 서비스, 호텔예약 서비스, 렌트카(rent car) 서비스 등을 묶어서 하나의 큰 서비스로 제공하는 것이다[1,2].

이러한 개별적인 웹 서비스의 조합을 지원하기 위하여 각 서비스를 표현하기 위한 WSDL(Web Service Description Language)이 개발[1,3]되었으며, 또한 개별적인 웹 서비스의 조합을 위하여 WS-BPEL(Web Service-Business Process Execution Language)이 개발[4,5,6]되었다. WS-BPEL은

※ 이 논문은 2011년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 연구되었음.
† 준 회 원 : 충북대학교 컴퓨터과학과 박사과정
†† 종신회원 : 충북대학교 소프트웨어학과 교수
논문접수 : 2012년 10월 15일
수 정 일 : 1차 2012년 11월 8일
심사완료 : 2012년 11월 19일
* Corresponding Author : Jang-Eui Hong(jehong@chungbuk.ac.kr)

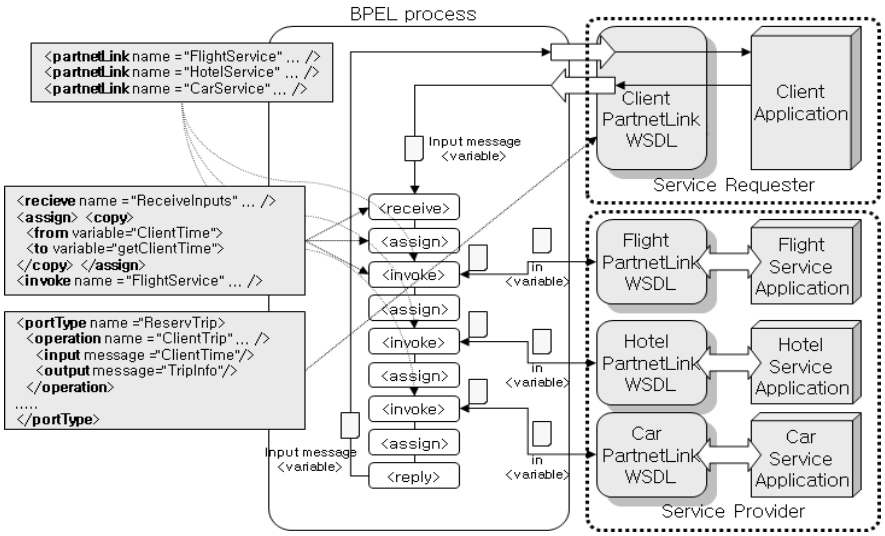


Fig. 1. WS-BPEL Specification for Travel Service

WSDL로 기술된 독립적인 웹 서비스를 요구사항에 맞는 서비스의 행위를 제공하기 위해 통합하는 SOA(Service-Oriented Architecture) 기반의 개발[1,4]에서 중요하게 사용된다. Fig. 1은 여행(Travel) 서비스를 제공하기 위한 각각의 항공, 호텔, 렌트카 서비스에 대한 WSDL 명세와 이를 조합하는 WS-BPEL의 명세를 보여준다.

이러한 웹 서비스의 조합을 통한 서비스 개발은 전체적으로 사용자의 요구사항을 만족하는 기능을 제공해야 함은 물론 성능 특히 응답시간 측면에서의 요구사항에 대해서도 만족해야 한다[7,8,9]. 즉, 항공과 호텔, 렌트카 서비스가 모두 성공적으로 이루어지는 경우에 사용자가 원하는 여행 서비스가 정상적으로 완료되는 것이며, 이 과정에서 각각은 사용자가 인내할 수 있는 수준에서의 성능, 예를 들면 대략적으로 1초에서 3초 이내의 응답시간 내에 서비스의 성공 또는 실패에 대한 결과를 제시할 수 있어야 한다. 3초 이상의 응답시간이 지나는 경우, 사용자는 일반적으로 버튼을 여러 번 클릭하여 2번 이상의 서비스를 실행시키거나 서비스 실행을 중지 시키는 원하지 않는 결과를 만들어 낼 수 있다.

따라서 조합된 서비스가 어느 정도의 성능을 제공할 수 있을 까에 대한 측정은 조합 서비스의 개발에 있어서 기능적 요구사항이외에 또 다른 하나의 중요한 서비스 요구사항이 되고 있다. 본 연구에서는 이러한 필요성에 따라 조합된 서비스를 테스트하기 위한 프레임워크를 제안한다. 제안하는 프레임워크에서는 조합된 서비스의 성능을 측정하기 위하여, WS-BPEL로 기술된 서비스의 프로세스 명세에 단위 서비스에 대한 실행 시간 측정을 위한 방법을 새롭게 도입하여 명세할 수 있도록 확장하였다. 즉, 실행 시간 측정을 위한 타이머 서비스를 Aspect 개념을 반영하도록 개발하고, 이를 기존 웹 서비스들과 조합함으로써, 단위 서비스에 대한 실행 시간 측정이 가능하도록 하였다. 또한 조합서비스의 성능을 테스트하기 위한 서비스 명세 및 측정도구인 PerToCS(Performance Tester to Composite Service)를 개발하였다. 본 논문에서 제시하는 테스트 프레임워크는 조합된 서비스가 사용자 감내 수준의 성능을 만족시키지 못하는 경우, 구성하는 단위 서비스를 동일한 다른 웹 서비스로 대체할 수 있도록 정보를 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 웹 서비스의 성능을 테스트하는 기존의 관련연구 및 본 연구와의 차이점을 분석하고, 3장에서는 성능 테스트를 위한 WS-BPEL의 확장에 대하여, 그리고 4장에서는 제안하는 성능 테스트 프레임워크에 대하여 설명한다. 5장에서는 예제를 통한 도구 활용과 성능 측정 실험에 대하여, 6장에서는 결론과 향후 연구내용을 기술하였다.

2. 관련 연구

2.1 기존 연구 분석

웹 서비스의 테스트에 대해서는 기존에 많은 연구들[7,8,10,11,12,13]이 있었다. 그러나 기존의 많은 연구들은 조합된 서비스가 올바른 수행 결과를 제공하는 가에 대한 기능적 측면의 테스트를 주로 다루고 있다. 본 연구에서는 성능 측면의 테스트에 주안점이 있기 때문에 이에 대한 기존의 연구들을 살펴보았다.

Bartolini[8]는 TCov라는 프로브(probe)를 이용하여 웹 서비스의 호출된 서비스 정보와 성능을 테스트하는 연구를 수행하였다. 개발자는 직접 서비스를 호출하고 실행되는 프로브를 통해 서비스를 테스트 하며, 테스트 중 자체 라이브러리에서 필요한 형태의 로그를 남기며 테스트 결과를 저장하도록 하였다.

Miller[7]는 웹 서비스 성능을 테스트하기 위하여 WSFL(Web Services Flow Language)를 사용하여 성능을 시뮬레이션하고 모니터링하는 SCET(Service Composition Execution Tool)를 제안하였다. 또한 Hao[11]는 에이전트 기

반의 성능 테스트 기법을 제안하였는데, 이는 테스트 경로 및 시나리오 생성기, 서비스 로드(Load) 생성 에이전트, 테스트 매니저 등의 포함하며 조합된 서비스보다는 단일 서비스의 테스트를 지원하도록 하였다.

위와 같은 기존의 연구들에서는 조합된 서비스의 성능 테스트를 위하여 WS-BPEL 엔진 자체를 확장해야 되며, 확장된 WS-BPEL 엔진에서만 사용할 수 있다는 활용상의 제약점을 가지고 있다. 프로브 기반의 테스트 역시, 프로브를 기반으로 하는 테스트 환경에서만 제약적으로 사용될 수 있다는 문제점이 있다. 모니터링 코드를 삽입하여 테스트하는 경우는 모니터링 코드를 테스트 대상이 되는 모든 서비스의 응용 어플리케이션에 포함시켜야 한다. 이처럼 기존 연구들은 특정 인프라가 필요하며, 이를 제어할 수 있는 테스트 프레임워크가 필요하기 때문에 성능 테스트에 대한 많이 비용이 수반된다. 또한 시뮬레이션을 통한 성능 테스트가 이루어지기 때문에 정확한 성능을 테스트하기 어렵다는 문제점을 갖는다.

2.2 본 연구의 차별성

본 연구에서는 기존의 웹 서비스 성능 테스트처럼 프로브나 모니터와 같은 특정 인프라가 필요하지 않도록 관점지향 개발방식의 Aspect를 도입하였다. Aspect는 기존의 응용 코드를 수정하지 않고, 테스팅이 가능하다는 장점을 제공한다. 이러한 Aspect의 개념을 WS-BPEL과 연결하여 실제 배포된 응용 서비스의 성능을 테스트 하도록 연구하였다. 본 연구에서 제안하는 Aspect 기반의 성능 테스트 프레임워크가 제공하는 특징은 다음과 같다.

- 테스트 대상 범위의 유연성 : 관점지향의 특성상, 관심사 부분에 대해서만 테스트 수행이 가능하며, 테스트를 위한 기존 코드의 변경이 없다.
- 기존 BPEL 엔진과의 호환성 : 기존 엔진에 정확히 호환되며, 기존과 동일하게 사용하면서 지속적인 성능 로그를 받을 수 있다.
- 테스트 수행의 실제성 : 시뮬레이션을 통한 성능 측정이 아니라 서비스가 배포된 상태에서 실제 웹 서비스 호출하기 때문에 정확한 결과 값을 가져온다.
- 테스트 요구사항의 확장성 : Aspect 작성에 따라 성능 테스트 이외의 관심사에 대한 특정 부가기능을 쉽게 추가하여 테스트할 수 있다.

3. 실행 시간 측정을 위한 WS-BPEL의 확장

WS-BPEL은 정해진 절차에 의해 각각의 액티비티가 실행되고 <invoke> 액티비티에 의해 서비스의 호출이 일어난다[4]. WS-BPEL의 <invoke> 액티비티는 서비스 호출 후, 호출된 서비스로부터 회신될 때 까지 기다린다. 이것을 이용하여 호출 직전과 회신 시점을 기준으로 시간차이를 산출하여 서비스의 응답시간을 측정할 수 있다. 즉 WS-BPEL 명세에서 서비스가 호출(invoke)되기 직전에 타이머를 셋(set)하고 서비스 실행 후, 회신을 받는 시점에서 타이머를 정지시킴으로써, 응답시간을 산출하게 된다. 그러나 기존의 WS-BPEL에서는 이러한 타이머를 설정하기 위한 액티비티가 존재하지 않기 때문에 이에 대한 확장이 필요하다.

본 논문에서는 이러한 확장을 위하여 기존의 WS-BPEL에 대한 문법은 수정하지 않고 그대로 활용하기 위한 방법으로 관점 지향(Aspect-Orientation)에서의 Aspect[14]를 도입하였다. Fig. 2는 Aspect를 이용하여 타이머를 사용하는 개념을 제시한 것이다. 즉 횡단관심사가 될 서비스의 호출점인 <invoke>는 Aspect의 Pointcut으로써 WS-BPEL의 오퍼레이션과의 연결점을 나타내고 BeforeAdvice와 AfterAdvice는 각각 수행 시간 측정을 위한 타이머 서비스의 시작과 종료 역할을 수행한다. 이를 통합 조합된 단위 서비스의 응답 시간을 측정할 수 있게 된다. 개발된 Aspect 명세는 조합 서비스의 WS-BPEL 명세와 위빙(Weaving)되며, 해당 서비스의 <invoke> 액티비티를 Aspect의 Cut Points로 인식하게 된다. 이러한 인식은 사용자가 원하는 서비스의 실행과 무관하게 별도의 실행 과정을 거쳐 응답시간을 기록하게 된다.

Fig. 2와 같이 WS-BPEL 프로세스 명세에서 Aspect 컴포넌트를 사용하기 위해서는 WS-BPEL의 확장이 필요하다. 다시 말해서 본 연구를 통해 개발된 PerToCS의 도구에서 Aspect 컴포넌트의 명세 정보가 추후 위빙을 위하여 확장된 메타 모델에 저장될 수 있게 하기 위함이다. 이러한 확장에

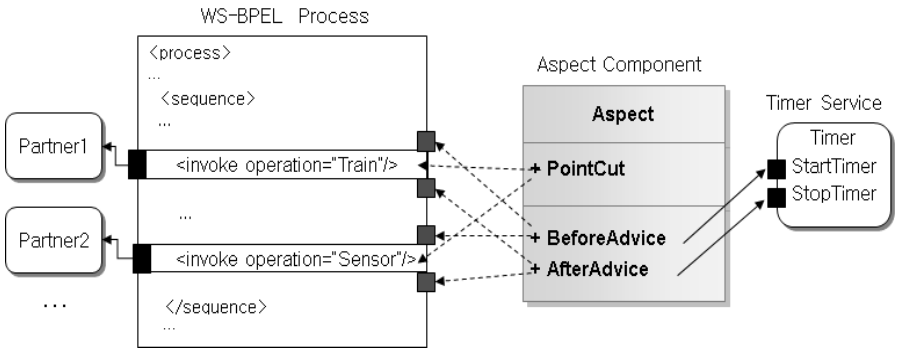


Fig. 2. Performance Measurement using Aspect

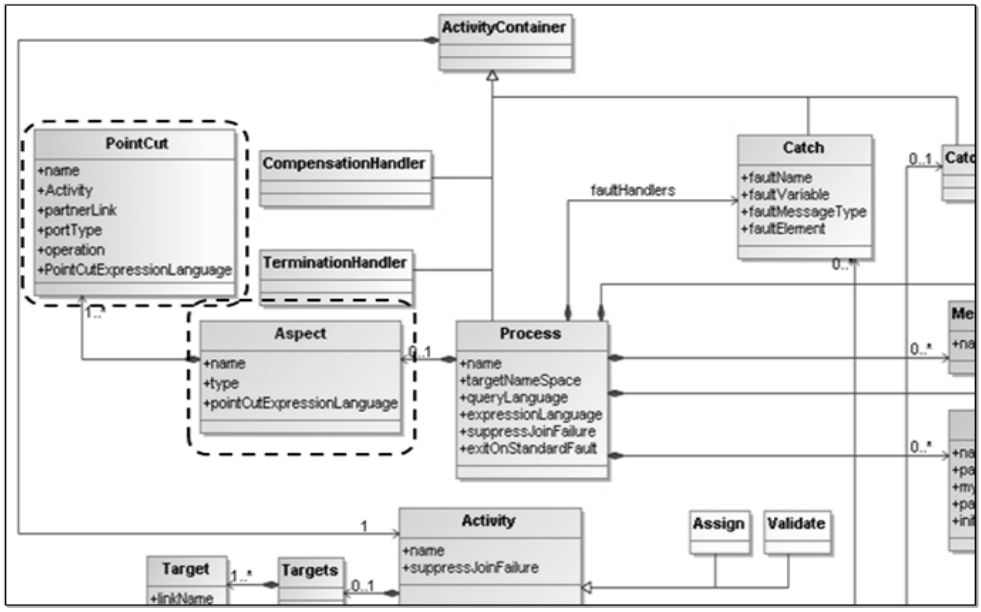


Fig. 3. Extended Meta-model of WS-BPEL

의해 정의된 WS-BPEL의 메타 모델[4]은 Fig. 3과 같다. Fig. 3은 확장된 일부만을 편집하였다.

4. 성능 테스트 프레임워크

본 논문에서 제시하는 조합 서비스의 성능 테스트 도구인 PerToCS는 NetBeans 6.5.1[15]을 이용하여 개발되었다. Netbeans 6.5.1을 사용하기 위해서는 선행적으로 JDk 버전 1.5를 설치해야 한다. 그림 4는 조합된 서비스의 성능 측정을 위해 구축된 성능 테스트 프레임워크의 전체적인 아키텍처를 보여준다. 그림 4는 웹 서비스 성능 테스트를 위한 주요 세 개의 컴포넌트인 PerToCS core, BPEL Designer, Glassfish V3내에 조합 서비스 명세, 서비스 위빙(Weaving), 서비스 호출 및 실행 등을 수행하는 기능들과 이들 컴포넌트간에는 데이터의 흐름(실선), 제어 메시지 흐름(점선) 등에 의해 상호 작용을 보여준다.

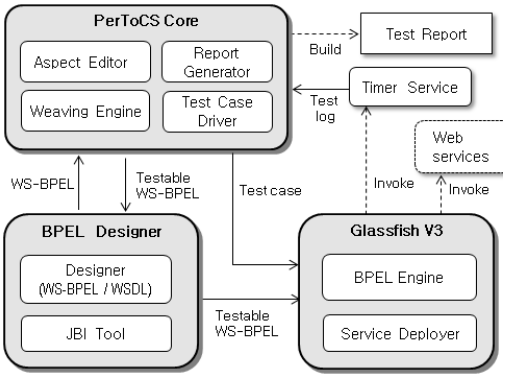


Fig. 4. Performance Test Framework Architecture for Composite Service

4.1 프레임워크 구성요소

1) BPEL Designer: BPEL Designer는 WSDL을 이용하여 웹 서비스를 기술하고, 이를 기반으로 서비스 조합을 가능하도록 하는 WS-BPEL에 대한 그래픽 기반 환경을 지원한다. JBI (Java Business Integration)는 기술된 웹 서비스 간에 전달되는 메시지들에 대하여 표준화된 메시지 전송 포맷에 맞도록 변환 및 해석을 담당한다[15,16]. BPE 디자이너는 WSDL이나 WS-BPEL에 대한 기능의 실행 흐름에 대한 비쥬얼한 명세를 가능하게 한다.

2) GlassFish V3: 이는 JEE5 플랫폼의 최신 기능을 구현한 오픈 소스용 응용 서버로써, 제시하는 프레임워크 상에서 조합된 웹서비스를 BPEL 엔진을 통해 실행하면서, 테스트를 위한 런 타임 환경을 제공한다[15].

3) PerToCS Core: PerToCS Core는 조합서비스의 실행 시간 측정을 위한 타이머 서비스를 Aspect 컴포넌트로 작성하는 편집기와 Aspect 컴포넌트와 조합 서비스 명세인 WS-BPEL을 위빙(weaving)하여 성능 테스트가 가능한 WS-BPEL 프로세스를 생성하는 역할을 수행한다.

또한 조합된 서비스가 실행될 때, 서비스 호출을 위해 요구되는 파라메터 데이터 - 여기서는 테스트 케이스 - 를 입력하고, 이로부터 서비스 실행 상태를 모니터링하여 결과 값을 출력하는 기능을 갖는다. 여기서 결과 값이란 서비스 실행에 따른 응답시간이다.

4) 기타: 본 논문에서 제시하는 성능 테스트 프레임워크의 기타 구성요소는 타이머 서비스와 테스트 리포터가 있다. 타이머 서비스는 Aspect 컴포넌트로부터 호출되어 서비스 실행 시간을 측정하여 로그를 남기는 기능을 수행하며, 테스트 리포터는 조합 서비스를 구성하는 각 서비스 단위별로 실행시간에 대한 결과 값을 사용자에게 전달하는 기능을 담당한다.

4.2 PerToCS

Fig. 4의 성능 테스트 프레임워크에서의 PerToCS Core는 본 연구에서 개발되어진 부분으로써, 이들에 대한 구체적인 지원 기능과 동작 과정은 다음과 같다.

1) Aspect Editor: 설치된 PerToCS에서는 WS-BPEL 코드를 입력 값으로 사용한다. Aspect의 PointCut은 모든 <invoke> 액티비티로 설정되어 있고, Advice에는 타이머 서비스의 호출과 종료에 대한 로직이 구현되어 있다. 그림 5와 같이 입력 또는 작성된 WS-BPEL 코드와 Aspect 작성 코드는 위빙을 수행하기 위한 버튼의 클릭을 통해 직조 과정이 수행된다. 수행된 직조 결과는 Fig. 5 화면의 오른쪽에 위치한 Weaved result 뷰에 나타난다.

2) Weaving Engine: 위빙 엔진은 WS-BPEL 코드와 Aspect 코드를 로딩하여 파싱(parsing) 과정을 통해 직조를 수행한다. 직조는 두 개의 코드 파일에 나타나는 동일한 웹 서비스에 대한 타이머 서비스의 연동과 이에 대한 시간 측정을 Testable WS-BPEL에 포함되도록 하는 과정이다. 생성된 Testable WS-BPEL은 BPEL Designer의 입력으로 사용된다. BPEL Designer로 표현된 직조된 결과는 Fig. 6과 같다. 이는 Fig. 5의 Weaved result에 대한 비쥬얼한 표현이다.

3) Reporter: 타이머 서비스는 테스트 수행시마다 서비스 응답시간을 체크하여 로그 파일을 생성한다. 테스트 결과로서의 저장된 로그 파일을 열면, Fig. 7과 같이 성능 테스트를 수행한 결과를 차트로 볼 수 있도록 제공하며, 각각의 서비스별 반응 시간과 평균 시간에 대한 정보를 직관적으로 알 수 있게 제공한다.

4) Test Case: PerToCS Core의 하위 기능으로 플러그 인 된 NetBeans의 Composite Application 메뉴를 이용하여 조합된 서비스를 테스트하기 위한 테스트 케이스를 생성할 수 있다. 생성된 테스트 케이스는 조합된 서비스를 실행시키기 위한 입력 데이터들로 구성되며, Fig. 4에 나타난 것과 같이 BPEL 엔진의 입력으로 사용되어 테스트가 수행된다.

5. 실험 및 분석

5.1 실험 설계

본 연구에서 개발한 조합 서비스의 성능 테스트 프레임워크, PerToCS를 이용하여 실험을 수행하였다. 실험은 통합된 여행 패키지 서비스이다. 이 서비스는 항공예약, 랜트카 예약, 그리고 호텔 예약의 독립된 3개의 서비스를 조합하고,

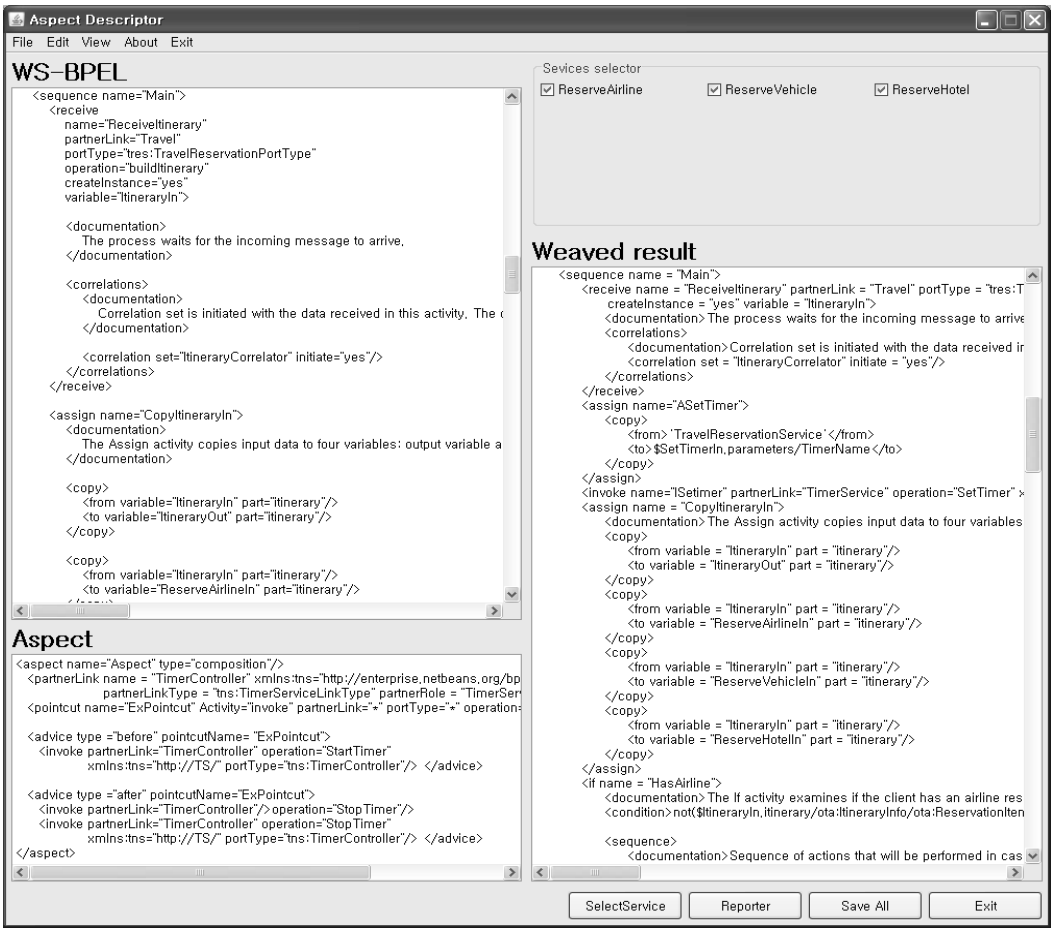


Fig. 5. Descriptor View of PerToCS Tool

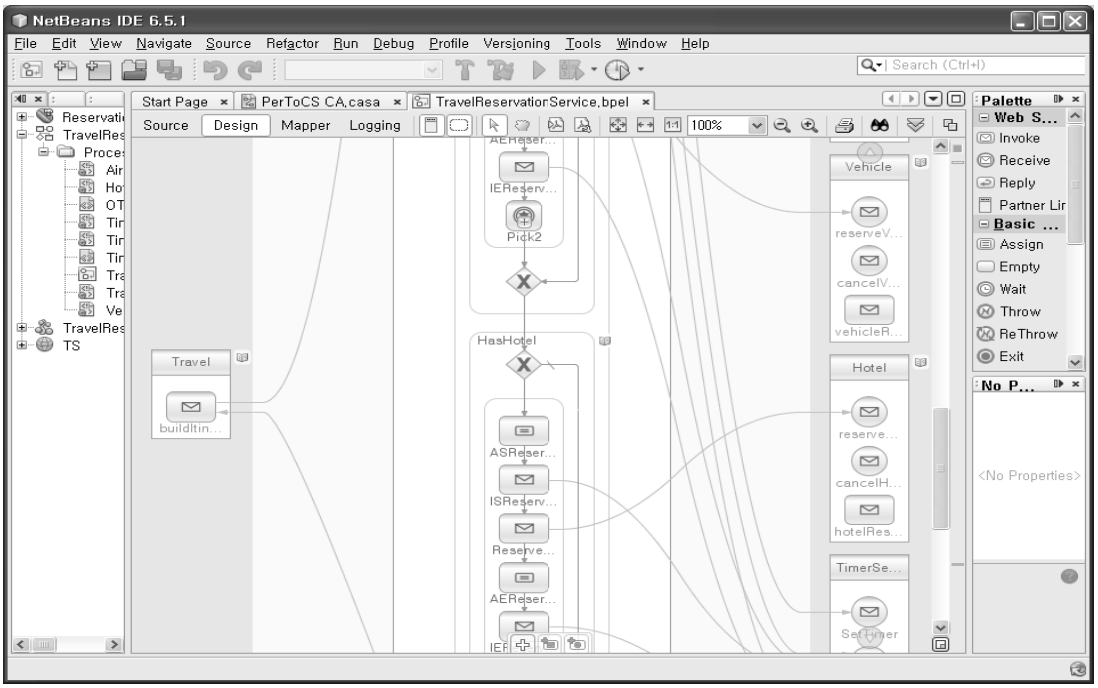
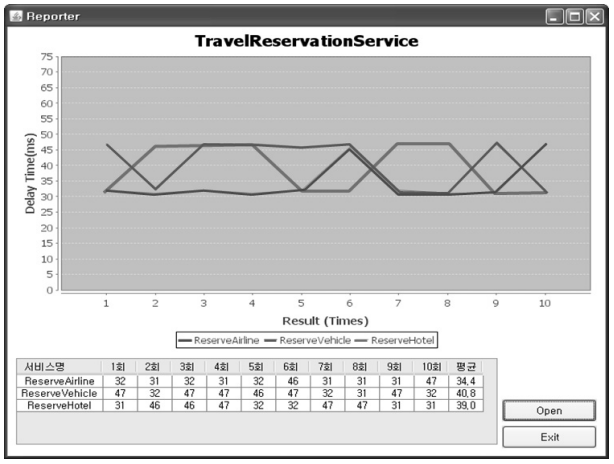


Fig. 6. A Screen of Weaving Result



서비스명	1회	2회	3회	4회	5회	6회	7회	8회	9회	10회	평균
ReserveAirline	32	31	32	31	32	46	31	31	31	47	34.4
ReserveVehicle	47	32	47	47	46	47	32	31	47	32	40.8
ReserveHotel	31	46	46	47	32	32	47	47	31	31	39.0

Fig. 7. Test Result Report Example of a Composite Service

이의 통합 서비스[17,18]를 위한 성능을 테스트하기 위한 것이다(Fig. 1 참조). 성능 테스트는 실제로 배포된 웹상의 서비스를 기반으로 테스트 되어야 하는 네트워크의 성능이나 부하에 따라 실행 시간의 차이가 발생할 수 있다. 따라서 가급적 네트워크의 사용량이 적은 임의로 시간에 인트라넷 환경에서 실험하였다. 실험은 다음의 형태로 진행하였다.

(1) 먼저 여행 패키지 서비스인 TBS(Travel Booking Service) 서비스의 성능을 측정하였다. TBS는 항공, 랜트카, 호텔 서비스의 조합 서비스이다.
(2) 조합서비스 TBS에 대하여 성능 테스트를 수행하고, 그 결과를 확인한다.
(3) TBS 테스트 결과, 응답 성능이 저조한 서비스를 다른 단위 서비스로 대체한다.
(4) 다른 단위 서비스로 대체한 조합 서비스를 TRS(Travel Reservation Service) 정의한다.
(5) 조합 서비스 TRS에 대하여 성능 테스트를 수행한다.
(6) TBS 서비스와 TRS 서비스의 성능 테스트 결과를 비교한다.

5.2 시험 환경 셋업

성능 테스트를 위하여 실험 수행에 앞서 조합 서비스의 배포와 같은 선행적 환경 설정 작업이 요구된다. 조합 서비스를 실행하기 위해서 먼저 기 개발된 타이머 서비스를 배포해야 한다. 또한 Fig. 8과 같이 넷 빈에서 새로운 프로젝트 TRS를 생성한다. 생성된 프로젝트는 모든 TRS에 대한 WSDL, WS-BPEL, 파트너 서비스 및 배포를 위한 Composite Application 등이 프로젝트에 로드되어 배포된다.

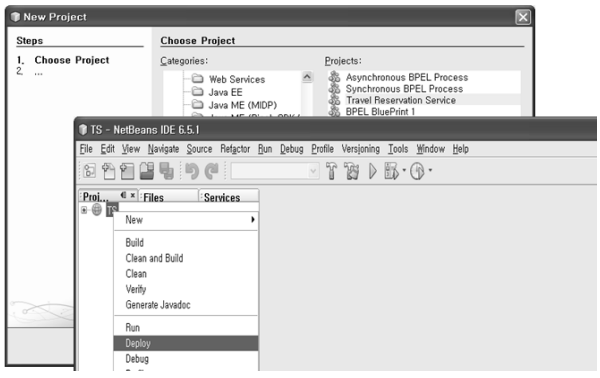


Fig. 8. Project Creation and Service Deploy for TRS

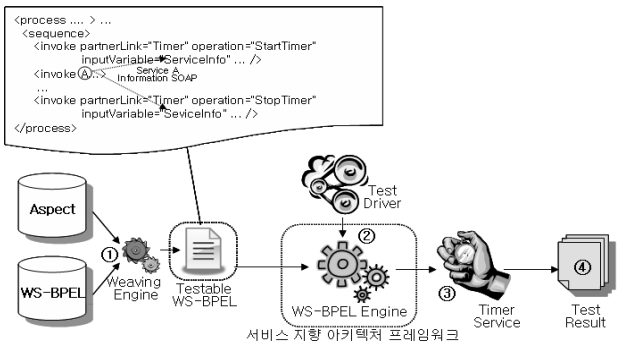


Fig. 9. Procedure of Performance Testing

5.3 실험 절차

테스트를 위한 프로젝트가 구성되었으면, 조합 서비스의 성능을 테스트하기 위한 과정이 진행될 수 있다. 본 논문에서 제시하는 프레임워크를 기반으로 조합서비스에 대한 성능 측정을 위한 테스트 수행 절차는 Fig. 9와 같다.

즉, 타이머 서비스 컴포넌트와 조합 서비스를 명세한 WS-BPEL을 위빙한 후에 테스트 드라이버(Test Driver)를 통해 테스트 데이터를 입력한다. 입력 데이터를 파라메터로 이용하여 조합 서비스를 구성하는 각 단위 서비스인 항공예약, 렌트카 예약, 호텔예약 서비스가 순차적으로 실행되고, 각 서비스 실행 단위별로 실행 시간이 측정된다. Fig. 9에서 테스트 드라이버에 의해 입력되어지는 테스트 데이터는 BPEL 엔진에 XML 형태로 입력되며, 입력데이터의 일부는 Table 1과 같다.

5.4 실험 결과

Table 1에 주어진 입력 데이터에 대하여 먼저 TBS에 대한 성능 즉 서비스 응답시간의 측정하였다. 실험은 총 10회 수행되었으며, 이의 평균 응답시간(ART)을 산출하였고, 결과는 Table 2와 같다.

Table 2에서와 같이 ReserveAirline과 ReserveHotel에 비해 ReserveCar는 상대적으로 높은 실행 시간을 보였다. ReserveCar의 지연은 전체 조합된 서비스의 지연을 초래한다. 따라서 렌트카를 예약하기 위한 단위 서비스를 다른 서비스(ReserveVehicle)로 대체한 TRS 서비스에 대하여 성능 테스트를 재 수행하였으며, 결과는 Table 3과 같다.

조합 서비스 TBS와 TRS에 대한 성능 테스트 수행 결과를 요약하면 Table 4와 같다. 전체적으로 TRS 서비스에서

Table 1. Test Data for the Composite Service

정보유형	입력 데이터의 XML 명세
개인정보	- <PersonName> <NamePrefix>**Mr.**</NamePrefix> <GivenName>**Robert**</GivenName> <MiddleName>**Anthony**</MiddleName> <Surname>**Jones**</Surname> </PersonName> <Telephone PhoneNumber="**2241630**" AreaCityCode="**302**" Extension="**5574**" /> <Email>**rajones@somewhere.org**</Email> - <Address FormattedInd="**true**"> <StreetNmbr PO_Box="**P.O. Box 77**">**1452 S. 13th St. N.W.**</StreetNmbr> <CityName>**Westfinster**</CityName>
항공예약	- <Air DepartureDateTime="**2003-03-03T08:00:00**" FlightNumber="**942**" ResBookDesigCode="**Y**"> <DepartureAirport LocationCode="**DEN**" /> <ArrivalAirport LocationCode="**ORD**" /> <MarketingAirline>**UA**</MarketingAirline> - <Seats> <Seat Number="**07A**" Characteristic="**Window**" CustomerRPH="**01**" /> </Seats> </Air>
렌트카 예약	- <Vehicle> <ConfID Type="**14**" ID="**02138525448412**" /> - <VehRentalCore PickUpDateTime="**2003-03-03T15:30:00**" ReturnDateTime="**2003-03-10T06:00:00**"> <PickUpLocation LocationCode="**YUL**">**Montreal Dorval Airport**</PickUpLocation> <ReturnLocation LocationCode="**YUL**">**Montreal Dorval Airport**</ReturnLocation> </VehRentalCore> - <Vehicle AirConditionInd="**true**" TransmissionType="**Automatic**"> <VehType VehicleCategory="**CAR**" DoorCount="**4**" />
호텔예약	- <RoomTypes> <RoomType RoomTypeCode="**KNGSTE**" NumberOfUnits="**1**" /> </RoomTypes> - <RoomRates> - <RoomRate> - <Rates> - <Rate AgeQualifyingCode="**10**" MinAge="**18**" EffectiveDate="**2003-03-03**" ExpireDate="**2003-03-10**"> <Base AmountAfterTax="**1495.00**" CurrencyCode="**CAD**" />
지불정보	- <FormOfPayment> - <PaymentCard CardNumber="**3151002645983754**" ExpireDate="**1205**" CardType="**MC**"> <CardHolderName>**ROBERT A. JONES**</CardHolderName> </PaymentCard> </FormOfPayment> <CostTotals AmountAfterTax="**1957.92**" CurrencyCode="**USD**" /> </TravelCost> - <UpdatedBy> <Access ID="**U09932147**" />

Table 2. Response Time of TBS Service (ms)

	1	2	3	4	5	6	7	8	9	10	ART
ReserveAirline	47	32	31	32	32	31	32	31	32	32	33.2
ReserveCar	56	67	57	72	57	59	63	56	57	68	61.2
ReserveHotel	47	46	31	31	31	47	47	47	31	31	38.9

Table 3. Response Time of TRS Service (ms)

	1	2	3	4	5	6	7	8	9	10	ART
ReserveAirline	32	31	32	31	32	46	31	31	31	47	34.4
ReserveVehicle	47	32	47	47	46	47	32	31	47	32	40.8
ReserveHotel	31	46	46	47	32	32	47	47	31	31	39.0

렌트가 예약을 위한 응답시간은 TBS의 렌트카 서비스 보다 33% 감소되었으며, 전체 조합 서비스 측면에서도 TRS가 14.3% 정도의 빠른 서비스가 제공될 수 있음을 확인하였다.

Table 4. Performance Comparison of TBS, and TRS

서비스명	TBS	TRS	수행시간차	증감량(%)
렌트카 서비스	61.2	40.8	-20.4	-33.33
전체 조합	133.3	114.2	-19.1	-14.33

5.5 실험 결과 분석 및 토의

앞서 본 논문에서는 항공, 차량, 호텔의 예약 서비스를 조합한 통합 여행 패키지 서비스에 대한 응답시간에 대한 측정 실험을 수행하였다. 앞선 수행한 실험에 추가하여 Apect 기반의 서비스 성능 측정 방법에 대한 유효성을 확인하기 위하여 추가적인 실험을 수행하였다.

실험 대상 서비스는 Purchase-Order 조합 서비스[19]이며, 조합된 서비스는 Order 서비스, Payment 서비스, Shipping 서비스로 구성되어 있다. Order 서비스는 고객이 구매할 상품에 대한 정보를 입력 받아 주문, 처리하는 서비스이고, Payment 서비스는 주문된 상품들의 가격을 계산하고 결재하는 서비스이다. Shipping 서비스는 배송 날짜를 결정하는 역할을 수행한다. 여행 패키지 서비스에 대한 실험과 마찬가지로 총 10회 실험을 수행하였으며, 결과는 Table 5와 같다.

Table 5에서 보는 것과 같이 Payment 서비스가 다른 2개의 서비스보다 상대적으로 높은 응답 시간을 보였다. 따라서 조합 서비스의 전체적인 응답시간의 감소를 위해 우선적으로 Payment 서비스의 대체를 고려할 수 있다.

위와 같은 실험을 통하여 우리는 다음과 같은 실험에서의 고려사항을 정리하였으며, 이러한 고려사항은 본 실험의 당위성을 설명할 수 있다고 판단한다.

- 서비스의 응답시간을 측정하기 위해 요구되는 오버헤드가 서비스 실행 시간에 영향을 줄 수 있다는 측면이다. 본 연구에서는 Aspect 기반의 응답시간 측정 메커니즘을 사용함으로써, 각 서비스 프로세스의 실행과 무관하게 동작한다. 따라서 응답시간 측정이 각 서비스의 실행 시간에 영향을 주지 않는다고 할 수 있다. 따라서 각 서비스에 대한 응답시간의 합은 전체 서비스의 응답시간으로 고려할 수 있다.
- 본 연구에서 수행한 실행 시간 측정이 인터넷 서비스의 실행 요청을 통해 측정된 것이므로, 서비스가 배치된 네트워크 환경이나 서버의 부하에 의존적이며 실험 결과가 매번 다를 수 있다는 것이다. 이러한 점을 극복하기 위하여 가급적 부하가 적은 시간에 실험을 수행하였으며, 또한 어느 시점이든 네트워크나 서버의 부하는 예상할 수 없는 것이기 때문에 이들을 모두 고려한 실험을 수행하는 것이 실제 사용자 측면에서의 응답시간에 해당된다고 판단하였다.
- 본 연구의 실험결과에 대한 활용은 각 서비스에 대한 사

Table 5. Response Time of Purchase-Order Service (ms)

	1	2	3	4	5	6	7	8	9	10	ART
Order	23	25	23	29	25	27	32	27	27	28	26.6
Payment	48	48	50	44	48	47	47	49	45	47	47.3
Shipping	31	32	31	28	29	34	33	33	32	31	31.4

용자 응답시간을 제공하는데도 의미가 있지만, 그 보다 더 중요한 것은 실험을 통해 조합 서비스 중에서 응답시간이 오래 걸리는 서비스를 다른 서비스로 대체할 수 있도록 정보를 제공하는데 있다.

6. 결 론

WS-BPEL은 인터넷에서 유용한 단위 서비스들을 조합하기 위해 제공되는 서비스 실행의 명세 언어이다. 이러한 명세언어를 이용하여 보다 큰 서비스의 개발이 가능해졌으며, 다양한 서비스의 조합으로 인하여 서비스 실행의 기능적, 성능적 측면의 신뢰성 확보가 중요해졌다.

본 연구에서는 WS-BPEL 언어를 이용한 조합서비스의 실행 과정에서 사용자 서비스의 편의성을 점검하기 위한 서비스 실행 성능 테스트 기법을 제안하였다. 제안된 기법은 Aspect 컴포넌트로 구현된 타이머 서비스의 명세를 기존의 WS-BPEL 명세와 위빙하여 BPEL 엔진에서 실행 하면서, 단위 서비스의 실행시간에 대한 측정을 가능하도록 한다. 이러한 성능 테스트의 지원을 위하여 NetBeans IDE 6.5.1을 이용하여 성능 테스트 프레임워크인 PerToCS 도구를 개발하였다.

본 연구를 통해 제시한 성능 테스트 프레임워크는 조합된 서비스의 전체적인 성능을 향상시키기 위하여 성능이 낮은 구성단위 서비스를 대체할 수 있도록 서비스 응답시간 정보를 개발자에게 제공하며, 이를 통해 서비스 기반 개발에서의 성능 향상을 위한 테스트 기법으로 활용될 수 있다. 향후의 연구는 성능 테스트 이외에 서비스 안전성 등과 같은 서비스 품질 요소를 점검할 수 있는 기법을 개발하는 것이다.

참 고 문 헌

[1] T. Erl, *Service-Oriented Architecture : Concepts, Technology, and Design*, Prentice Hall, 2005.

[2] N. Komoda, "Service Oriented Architecture (SOA) in Industrial Systems", *Proceeding of IEEE International Conference on Industrial Informatics*, pp.1-5, 2006.

[3] W3C, "Web Services Description Language (WSDL) Version 2.0", TR: EC-wsdl20- 20070626. http://www.w3.org/TR/wsdl20/, June 2007

[4] OASIS, "WS-BPEL 2.0 (Web Services-Business Process Execution Language)", http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007.

[5] F. Leymann, *Web Services Flow Language (WSFL 1.0)*, IBM Software Group, 2001.

[6] S. Thatte, *XLANG - Web Service for Business process design*, Microsoft Corporation, 2001.

[7] J. A. Miller et al., "Performance Analysis and Simulation of Composite Web Services", *Journal of Electronic Markets*, Vol.13, No. 2, pp.120-132, 2003.

[8] C. Bartolini, et al., "Whitening SOA testing", *Proceedings of the 1st international workshop on Quality of service-oriented software systems*, 2009.

[9] S.H. Kuk and H.S. Kim, "Automated Functionality Test Methods for Web-based Applications," *KIPS Transaction, 14D-5*, pp.517-530, 2007

[10] T. T. Cheng and C. H. Fu, "On the Development of Software Tools for Testing Web Services", *Proceedings of International Conference on Internet Computing*, 2004.

[11] D. Hao, Y Chen, F. Tang, F. Qi, "Distributed agent-based performance testing framework on Web Services" *Proceedings of Software Engineering and Service Sciences (ICSESS),* 2009.

[12] J. Huang, Y. Gong, "An EMF Activity Tree Based BPEL Defect Pattern Testing Method" *Proceeding of International Conference on Computer Engineering and Technology*, 2010.

[13] T. Lertphumpanya, T. Senivongse, "A basis path testing framework for WS-BPEL composite services", *Proceedings of the 7th WSEAS*, 2008.

[14] G. Kiczales, et al., "An Overview of AspectJ", *LNCS, Vol. 2072,* Springer, pp.327-353, 2001.

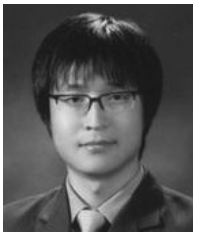
[15] NetBeans, "NetBeans IDE 6.5.1 Download," http://netbeans.org/downloads/6.5.1/

[16] Do van Thanh, Ivar Jorstad, "A Service-Oriented Architecture Framework for Mobile Services" *Proceedings of the Advanced Industrial Conference on Telecommunications*, 2005.

[17] Gopalan S. R. "JBI/SOA Blueprints: Travel Reservation Scenario" http://blogs.oracle.com/gopalan/entry/jbi_soa_blueprints_travel_reservation, 2006.

[18] Open Travel Alliance, "Open Travel Schema", http://www.opentravel.org/OTA/2003/05, 2003.

[19] OASIS, Advancing open standards for the information society, http://www.oasis-open.org/apps/org/workgroup/wsbpel/, 2011.



김 종 필

e-mail : kimjp@selab.cbnu.ac.kr

2006년 충북대학교 컴퓨터공학과(학사)

2008년 충북대학교 컴퓨터과학과(석사)

2008년~현 재 충북대학교 컴퓨터과학과 박사과정

관심분야 : Aspect programming, Embedded software, Software Quality engineering, Service-oriented architecture

홍 장 의

e-mail : jehong@chungbuk.ac.kr
1988년 충북대학교 전산학과(학사)
1990년 중앙대학교 컴퓨터공학과(석사)
2001년 KAIST 전산학과(박사)
2002년 국방과학연구소 선임연구원
2004년 (주)솔루션링크 기술연구소장
2004년~현 재 충북대학교 소프트웨어학과 교수
관심분야 : Software Engineering, Embedded software, Quality-based software engineering, Modeling and verification, Low-power software development