위성항법 시뮬레이션 작업을 동적으로 지원하는 테스트 프레임워크☆*

A Test Framework for Dynamically Supporting the Simulation Works of the Global Navigation Satellite Systems

국 승 학* 김 현 수** 이 상 욱***
Seung Hak Kuk Hyeon Soo Kim Sanguk Lee

요 약

시뮬레이션은 어떤 문제를 모의적으로 실험하여 그 특성을 파악하는 작업이다. 시뮬레이션 과정에서는 시뮬레이션 모델, 알고리즘, 입출력 데이터의 교체 및 변경이 빈번하게 발생한다. 특히 알고리즘의 교체를 통한 시뮬레이션 작업의 경우 알고리즘을 구현한 컴포넌트가 교체될 때 기능적으로 정확하게 동작하지 않는다면 시뮬레이션 작업을 성공적으로 수행하기 어렵다. 이 논문에서는 소프트웨어 기반 위성항법 시뮬레이션 환경에서 교체될 컴포넌트가 기능적으로 정확하게 구현되어 있는지를 검증하기 위한 테스트 프레임워크를 제안한다. 이 프레임워크는 컴포넌트가 교체되는 시점에서 교체되는 컴포넌트의 상황에 맞게 기능 테스트를 수행할 수 있게 해준다.

ABSTRACT

Simulation is the work that identifies the characteristics of some problem through the simulated experiments. During the experiments it is frequently required to change or replace the simulation models, algorithms, or input/output data. Especially, in the case of the simulation works performed by replacing algorithms, if a replaceable component that implements a specific algorithm is not correct with respect to its functionality it is very difficult to carry out the simulation works successfully. In this paper, we suggest a test framework that verifies functional correctness of the replaceable component in the software-based GNSS (Global Navigation Satellite System) simulation environments. When a component is replaced, this framework enables us to properly execute the functional test for the component according to its context.

☞ KeyWords : Global Navigation Satellite System (GNSS), Simulation, Software Testing, Functional Testing, 위성항법 시스템, 시뮬레이션, 소프트웨어 테스팅, 기능 테스트

1. 서 론

시뮬레이션이란 복잡한 문제를 해석하거나 사회현상 등을 해결하기 위하여 실제 또는 가상의 시스템 모형 또는 모델을 만들어 모의적으로 실험하여 그 특성을 파악하는 작업이라 정의할 수 있다[1]. 이러한 시뮬레이션은 실제 시스템의 동작에 대한 정보를 획득하거나 시스템의 성능 향상을 위해 운영 방법 및 자원 관리 방법의 개발, 새로운 개념 및 시스템의 구현 전 테스트, 시스템의 구축 전 동작에 대한 정보 획득을 위해 널리 사용된다.

시뮬레이션 작업은 다양한 시뮬레이션 모델의 구축 및 검증 작업을 위해 시뮬레이션 모델 자체의 변경, 특정 알고리즘의 교체, 시뮬레이션 과정

* 정 회 원 : 충남대학교 컴퓨터공학 박사 재학
triple888@cnu.ac.kr
** 정 회 원 : 충남대학교 전기정보통신공학부 교수
hskim401@cnu.ac.kr
*** 정 회 원 : 한국전자통신연구원 위성관제-항법연구팀 책임연구원 slee@etri.re.kr
[2009/05/04 투고 - 2009/05/12 심사 - 2009/06/05 심사완료]
☆ 본 연구는 지식경제부 및 산업기술연구회의 협동연구과제의 일환으로 수행하였음. [08AR2310, GPS/Galileo환경에서의 위성항법신호생성/수신처리 및 측위성능향상 기초연구]

에서의 입출력 데이터의 변경과 같은 수정에 대한 요구사항이 빈번하게 발생한다. 특히 안정된 시뮬레이션 모델 위에서 다양한 알고리즘의 교체를 통해 시뮬레이션 작업을 수행하는 경우에는 알고리즘들을 컴포넌트로 구현하여 그 컴포넌트들을 교체해 가면서 어떤 효과가 발생하는지 관찰한다. 이러한 상황에서 교체될 컴포넌트가 기능적으로 정확하게 동작하지 않는다면 목적했던 시뮬레이션을 성공적으로 수행하기 어려울 것이다. 따라서, 시뮬레이션을 수행하기 이전에 교체될 컴포넌트가 기능적으로 정확하게 구현되어 있는지를 검증하는 일은 시뮬레이션의 정확도와 시뮬레이션의 수행시간에 절대적으로 영향을 미친다.

교체될 컴포넌트를 위한 기능 테스트를 시뮬레이션 시스템 전체의 관점에서 수행한다면 많은 비용이 소요될 것이다. 따라서 교체되는 컴포넌트 상황에 적절한 범위에서 기능테스트가 수행되어야 한다. 그러나 기존의 시뮬레이션 알고리즘의 테스트 방법은 대부분 시스템 전체를 대상으로 하는 기능 테스트 방법으로 수행되기 때문에 최종 시스템 테스트를 수행하기 이전에 개별 컴포넌트의 결과가 정확한지 알기 어렵고, 많은 시간과 비용이 요구된다.

특히 위성항법 시뮬레이션 시스템의 경우 시뮬레이션 알고리즘의 변경, 교체에 대한 요구사항이 많기 때문에 적절한 범위의 컴포넌트 테스트 방법이 필요하다. 지금까지의 위성항법 시뮬레이터는 기존의 GPS (Global Positioning System) 기반 시뮬레이션 기능을 수행하였다. 그러나 향후 개발될 위성항법 시뮬레이터는 GPS뿐만 아니라 현대화된 GPS, 갈릴레오(Galileo) 위성의 시뮬레이션 기능을 포함해야 한다. 현재 이와 관련된 다양한 시뮬레이션 알고리즘이 개발되고 있는 상황이며, 이들의 변경 및 교체를 통한 시뮬레이션 작업의 수행 요구가 빈번하게 발생하고 있다. 그러므로 개발되는 다양한 알고리즘을 적은 비용과 노력으로 테스트하는 방법이 필수적이다. 이런 목적을 위해 본 연구에서는 소프트웨어 기반 위성항법 시뮬레이터에서 시뮬레이션 알고리즘이 교체되는 시점에서 교체되는 컴포넌트(알고리즘 구현)에 초점을 맞춰 수행할 수 있는 기능 테스트 방법을 제안하고 그것을 수행하기 위한 테스트 프레임워크 및 이 때 적용될 테스트 데이터와 테스트 오라클 생성 방법을 제안한다.

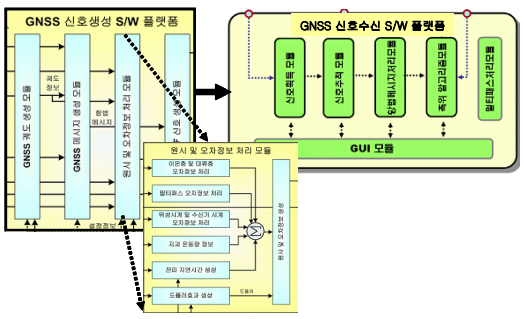
2. 관련 연구

2.1 시뮬레이션 시스템의 분류

시뮬레이션 시스템은 분류 기준에 따라 매우 다양하게 분류된다. [2]에서는 병렬 및 분산시스템에 대한 시뮬레이션 방식의 다양한 분류 기준을 제시하고 있다. [2]에 의하면 시뮬레이션 작업은 시간적인 측면에서 정적(static) 및 동적(dynamic) 방식으로, 적용되는 값을 기준으로 이산(discrete) 및 연속(continuous) 방식으로, 행위(behavior)적인 측면에서 결정적(deterministic) 및 확률적(probabilistic) 방식으로 분류된다. 본 논문에서는 다음과 같은 세 가지 분류 기준에 따라 시뮬레이션 시스템을 분류한다. 첫째는 입력 데이터의 변화를 통한 시뮬레이션 수행 방식이다. 이 방식에서는 입력 데이터 값이나 입력 이벤트에 변화를 주고 그 효과를 관찰하는 것이 목적이다. 이 경우에는 시뮬레이션 모델이 고정되어 있다. 두 번째 방식은 시뮬레이션 모델 구성요소의 알고리즘 변화를 통한 시뮬레이션 수행 방식이다. 이 방식의 경우 동일한 입력에 대해서 교체된 알고리즘의 효과를 관찰하는 것이 목적이다. 세 번째 방식으로는 프로세스의 변경을 통한 시뮬레이션 방식이다. 모델링하고자 하는 시스템 구성 요소의 알고리즘 수준의 변화가 아니라 여러 요소로 구성되는 프로세스의 변화 즉 프로세스에 새로운 활동이 추가되거나 삭제되거나 교체되는 변화에 대해 그 효과를 관찰하는 방식이다.

2.2 소프트웨어 기반 위성항법 시뮬레이터

그림 1은 위성항법 시뮬레이터의 개략적인 구조이다[3]. 위성항법 시뮬레이터는 갈릴레오 및 GPS 항법신호를 생성하는 신호생성기와 이들 신호를 처리하는 신호수신기로 나뉘어 진다. 신호생성기는 실제 GPS/갈릴레오 위성에서 생성하는 신호를 소프트웨어적으로 생성하는 기능을 수행한다. 또한 지구를 둘러싼 대기권 및 이온층과 같은 환경적인 요인으로 인해 발생하는 다양한 현상을 반영하여 실제 신호와 유사한 신호를 발생시킨다. 신호수신기는 수신된 신호를 처리하기 위해 신호를 획득하고 추적하여 네비게이션 시스템과 같은 사용자 애플리케이션에서 활용될 수 있는 항법메시지와 측정값을 추출하는 기능을 수행한다.



(그림 1) 위성항법 시뮬레이터의 구조

위성항법 시뮬레이터는 위성항법 단말기와 서비스 개발을 위한 인프라로서의 역할을 수행한다. 즉 사용자 단말기와 위성항법 응용분야 시스템의 주요 알고리즘들의 기능 및 성능 평가를 위한 개발/시험환경을 제공한다. 이러한 목적을 위해서 시뮬레이터는 소프트웨어 기반으로 설계되고 구현되어야 한다.

2.3 컴포넌트 교체에 적용되는 테스트 기법

시뮬레이션 시스템의 테스트 방법은 적용하는 시뮬레이션 방식에 따라 달라 질 수 있다. 이 논문에서는 시뮬레이션 모델 구성 컴포넌트의 알고리즘의 변화를 통한 시뮬레이션 수행 방식에서 구성 컴포넌트의 교체에 대한 테스트 방법을 제시하고 있다. 그런데 컴포넌트 교체에 대한 테스트 과정에는 다음과 같은 테스트 방법들을 적용할 수 있다.

첫째, 적합성 테스트 방법을 적용할 수 있다. 적합성 테스트는 컴포넌트가 주어진 명세나 표준을 준용하는가를 시험하는 방법이다. 기본적으로 교체될 컴포넌트는 원래의 컴포넌트와 인터페이스가 일치하여야 하므로 적합성 테스트를 통해 그것을 확인할 수 있다. [4]의 경우 컴포넌트의 상호운영성을 테스트하기 위해 개별 컴포넌트의 인터페이스와 엔티티 측면에서의 적합성 테스트 기법을 제시하였다. 일반적으로 컴포넌트는 자신이 제공하는 기능을 인터페이스로 공개한다. 이러한 인터페이스 테스트를 통해 그 기능이 요구사항과 일치하는지 테스트할 수 있다. [4]에서는 컴포넌트의 인터페이스뿐만 아니라 추가로 실제 컴포넌트가 갖는 개별 속성에 대한 검사를 통해 적합성을 테스트하는 방법을 제시하였으며, 이를 통해 각각의 컴포넌트의 상호운영성을 검사하였다.

둘째, 회귀 테스트 방법을 적용할 수 있다. 회귀 테스트는 어떤 컴포넌트가 수정되고 나면 그것이 원래 컴포넌트의 요구사항을 만족하는가를 시험한다. 여기서는 회귀 테스트 기법을 활용하여 새롭게 교체되는 컴포넌트가 기존 컴포넌트의 요구사항을 만족하는지 테스트 할 수 있을 것이다. [5] 논문은 컴포넌트 기반 시스템의 회귀 테스트 방법을 제안하였다. 여기서는 시스템을 구성하는 개별 컴포넌트의 회귀 테스트 모델을 구축하고, 주로 API의 변경에 초점을 맞춰 API 변경의 영향 범위를 자동으로 분석하여, 테스트하는 기법을 제시하였다. 그러나 좀 더 엄밀하게 보자면 회귀 테스트 기법을 적용하는 것이 적절하지 않을 수 있다. 회귀 테스트는 기존 컴포넌트에 변경이 가해졌을 경우 변경된 컴포넌트에서 변경된 내용과 그 영향 범위에 초점을 맞춰 테스트를 수행하게 되므로 이 논문의 경우처럼 기존 컴포넌트를 교체할 컴포넌트는 기존 컴포넌트로부터 변경된 것

이 아니라 독립적으로 새롭게 작성된 컴포넌트이기 때문에 회귀 테스트 방법을 곧바로 적용하기에는 개념적으로 일치하지 않는 부분이 존재하게 된다.

셋째, 기능(function) 테스트 방법을 적용할 수 있다. 기능 테스트 방법은 시스템이나 모듈이 의도했던 기능을 달성하고 있는지 여부를 시험하는 방법으로 시뮬레이션 시스템의 컴포넌트 테스트 방법으로 가장 적절하다. 이 논문에서의 시뮬레이션 시스템의 경우 동일한 기능을 실현하고 있는 여러 알고리즘들을 교체해 가면서 성능의 향상 정도를 평가하는데, 이런 목적을 위해서는 교체될 컴포넌트가 기존의 컴포넌트와 기능적인 측면에서 동일한 기능을 수행해야 한다는 개념을 전제로 갖고 있다. 따라서 교체될 컴포넌트와 기존 컴포넌트는 기능적인 측면에서 동일하되, 그 세부 구현이 다른 경우 이므로 교체될 컴포넌트는 기존 컴포넌트와 기능적인 측면에서 일치하는지 여부가 테스트되어야 한다. 그러나 일반적인 기능 테스트 방법은 널리 알려져 있으나 이 논문에서처럼 기존 컴포넌트와 그것을 대체하게 될 교체 컴포넌트 간의 기능의 일치 여부를 판단하기 위한 테스트 방법은 별로 제시되고 있지 않다. [6]에서는 기능적인 측면에서 그것이 동일한 기능을 수행하는지 여부를 판단하기 위한 기준을 제시하고 있다. 이 논문에서는 동일 기능을 수행하는지 판별하기 위한 기준으로 'Consistence with Purpose', 'Consistence within Product', 'Consistence with History', 'Consistence with Comparable Products' 등을 제시하고 있다. [7] 논문의 경우 우리의 의도에 근접한 접근 방법을 제시하고 있다. 이 논문에서는 J2ME(Java 2 Micro Edition) 라이브러리를 독자적으로 개발할 때 개발된 라이브러리가 표준에 적합한지 테스트하는 자동화된 도구를 제시하였다. 이 논문에서의 테스트 방법은 SUN사의 J2ME 라이브러리와 실제 개발한 라이브러리를 비교하여 SUN의 라이브러리를 대응 코드로 보고, 두 라이브러리 모듈의 수행 결과를 비교하는 방식으로 테스트를 수행하는 것이다. [8] 논문은 우리의 상황과 유사하게 시뮬레이션 환경에서의 기능성 테스트를 언급하고 있다. 포괄적 개념 측면에서는 우리가 제시한 방법과 비슷한 측면을 언급하고 있으나 우리 논문에서와 같이 테스트 프레임워크나 테스트 케이스 및 테스트 오라클의 획득 방법 등과 같이 구체적인 테스팅 요소들은 전혀 언급하고 있지 않다. 이 논문에서도 시스템 통합 테스트를 수행하기 이전에 시뮬레이션 시스템을 구성하는 기본 컴포넌트들에 대해 기능적으로 정확하게 구현되었는지 여부를 반드시 시험하기를 권하고 있다. [9] 에서는 시뮬링크(Simulink)라는 시뮬레이션 도구에서 컴포넌트들이 교체될 때 기능적 측면에서의 정확성을 검증하기 위해 확인 블록(verification block)을 제안하고 있다. 이 블록은 컴포넌트 테스트 케이스와 테스트 하니스(test harness)를 형성한다. 그러나 이 방법은 우리의 방법처럼 간결한 프레임워크가 아니고 어셜션 블록(assertion block)들을 조합하거나 확장하여 확인 블록을 만든다. 즉 검증하고자 하는 기능들에 대해 어셜션으로 표현하고 구현된 컴포넌트가 기능적인 측면에서 어셜션을 만족하는가를 확인함으로써 기능의 정확성 여부를 평가한다. 그러므로 기능을 검증하기 위해서는 사용자들이 각 기능마다 어셜션을 기술해줘야 한다. 그러나 우리의 방법은 교체될 컴포넌트가 기존 컴포넌트의 기능과 일치하는지 여부를 판단하기 위해 기존에 구축되어 있던 테스트 데이터와 테스트 오라클을 사용하여 컴포넌트의 수행 결과를 비교하는 방법을 사용하므로 사용자들이 개별 기능마다 각각의 어셜션을 기술하는 것과 같은 복잡한 작업을 수행하지 않아도 된다.

3. 위성항법 시뮬레이터의 테스팅

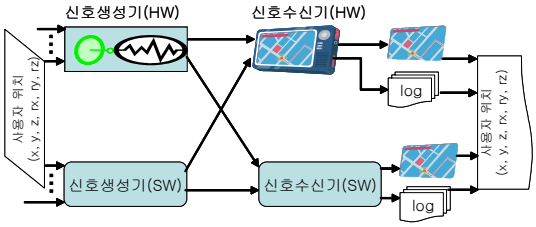
3.1 테스팅 관점

위성항법 시뮬레이터의 사용자들은 시뮬레이션 모델을 구성하는 다양한 알고리즘들을 컴포넌트

로 구현하여 그런 컴포넌트들을 교체해가면서 위성항법 애플리케이션 시스템의 기능적인 측면이나 성능적인 측면에서의 효과를 파악하고자 한다. 특히 현재의 GPS 위성 기반 위성항법 시뮬레이터에서는 향후 개발될 갈릴레오 위성 및 현대화된 GPS 위성 등과 같이 다양한 환경을 지원할 수 있는 기능이 요구되고 있다. 이 시뮬레이터는 다양한 시뮬레이션 알고리즘의 개발과 이들의 교체를 통한 향후 시스템의 예측, 시스템 구현 전 테스트, 교육 등 다양한 분야에서 활용될 것으로 예상된다. 따라서 위성항법 시뮬레이터는 지금까지 개발된 그리고 앞으로 개발될 다양한 알고리즘 컴포넌트의 변경/교체 기능이 필요하며, 이렇게 컴포넌트의 교체가 빈번하게 발생되는 상황에서 컴포넌트 교체 시, 우선 그 컴포넌트가 시뮬레이션 환경에 적합하게 설계되고 구현되었는지, 다음으로 그 컴포넌트가 다른 컴포넌트들과 자연스럽게 연동될 수 있는가에 대하여 테스트 되어야 한다.

3.2 시뮬레이터의 검증 환경

위성항법 시뮬레이터는 크게 신호생성기와 신호수신기로 구성된다. 지금까지의 신호생성기와 신호수신기에 대한 검증 방법은 세부 기능별로 기능이 정확하게 구현되었는지를 테스트하기 보다는 구현이 완료되고 나면 신호생성기, 신호수신기를 단위 대상으로 기능 테스트를 수행하여 그것들이 정확하게 구현되었는가를 판단하는 방식이었다.



(그림 2) 시뮬레이터 검증 환경

신호생성기와 신호수신기의 기능 테스트를 위한 검증 환경은 그림 2와 같다. 검증 환경은 하드웨어 신호생성기와 신호수신기를 근간으로 소프트웨어로 구현된 신호생성기와 신호수신기의 기능을 검증한다. 하드웨어 신호생성기는 실제 위성처럼 신호를 생성시키며, 이렇게 생성된 신호를 처리하는 하드웨어 신호수신기는 상업적인 신호수신기를 활용하기도 한다. 검증 방법은 크게 검증 모드와 시뮬레이션 모드로 나누어 수행된다.

검증 모드 : 신호생성기와 신호수신기의 기능 검증을 위한 과정으로 다음과 같이 진행된다.

- Shw → Rhw : 하드웨어로 구현된 신호생성기(Shw)와 신호수신기(Rhw) 간의 기능 검증
- Shw → Rsw : 검증된 하드웨어 신호생성기를 이용하여 소프트웨어 신호수신기(Rsw)의 기능 검증
- Ssw → Rhw : 검증된 하드웨어 신호수신기를 이용하여 소프트웨어 신호생성기(Ssw)의 기능 검증

기능 검증의 세부 절차는 다음과 같다.

1. 기본적인 입력 데이터와 함께 사용자의 위치 정보를 신호생성기에 입력으로 제공한다.
2. 신호생성기는 이러한 입력 정보를 가지고 여러 과정을 거쳐 최종적으로 IF(Intermediate Frequency) 신호를 생성한다.
3. 생성된 IF 신호는 신호수신기에 전달된다.
4. 신호수신기는 IF 신호를 자신의 처리 방식에 따라 내부적으로 처리한다.

5－1 신호수신기는 최종 결과인 사용자 위치 정보를 시각적인 방식으로 지도에 표시한다.

5－2 또는, 사용자 위치 정보를 로그 파일로 남긴다.

6. 신호수신기의 처리 결과인 사용자 위치 정보와 신호생성기에 입력으로 제공된 사용자 위치 정보를 비교한다.

입력으로 주어진 사용자 위치 정보와 결과로 나온 사용자 위치 정보가 일치하면 검증 대상 신호생성기 또는 수신기의 기능이 정확하다고 판단한다. 경우에 따라서는 두 데이터가 완전하게 일치하지 않을 수도 있다. 이것은 오차에 의해 발생하는 결과로 오차 인정 범위 내에 있으면 동일한 결과라고 판단한다. 어떤 환경에서는 하드웨어 신호생성기와 신호수신기를 대신하여 MATLAB으로 구현된 소프트웨어를 사용하여 검증하기도 하는데 그 기본 전략에는 차이가 없다. 신호생성기와 신호수신기의 정확도를 판단하기 위하여 이런 테스트를 충분하게 많이 수행하여야 하고 이 때 사용된 입력 데이터와 출력 결과들을 저장한다.

시뮬레이션 모드 : 소프트웨어 신호생성기와 신호수신기의 기능 검증을 마치고 나면 이를 바탕으로 적절하게 컴포넌트를 교체해 가면서 시뮬레이션 작업을 수행하는 과정이다.

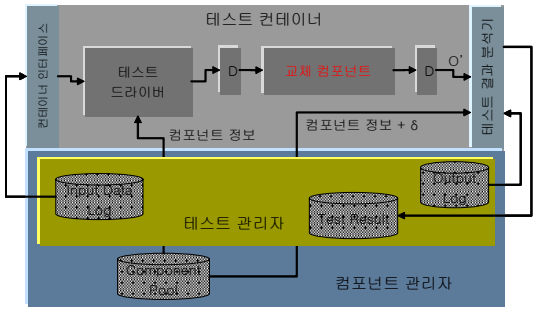
- Ssw → Rsw : 검증된 소프트웨어 신호생성기 및 신호수신기를 이용하여 시뮬레이션 수행

3.3 시뮬레이션 작업을 지원하기 위한 테스트 프레임워크

소프트웨어 기반 시뮬레이터의 경우에는 다양한 세부 기능들을 컴포넌트로 구현하여 그 기능들을 교체해 가면서 시뮬레이션 작업을 수행한다. 예를 들어, 위성항법 시뮬레이터의 신호생성기의 경우 위성의 궤도를 생성하는 모듈, 위성항법 메시지를 생성하는 모듈, 오류 정보를 생성하는 모듈 등이 있고 각 모듈 내부에는 다양한 세부 기능들이 존재한다 (그림 1 참조). 이러한 세부 기능들은 다양한 알고리즘의 컴포넌트로 구현되며, 이런 알고리즘들의 변경(즉 컴포넌트의 교체)을 통해 성능의 변화를 시뮬레이션 할 수 있다.

그림 2는 신호생성기 또는 신호수신기 단위에서 기능 테스트를 위한 검증 환경이므로 교체될 세부 기능 단위의 컴포넌트에 대한 기능 테스트를 수행하기에는 효율적이지 않다. 왜냐하면 하나의 세부 기능 컴포넌트가 교체된 경우에도 전체 생성기 혹은 수신기의 기능 테스트를 수행하고, 그 과정을 통해 교체된 컴포넌트의 기능의 정확성 여부를 판단하여야 하므로 많은 시간과 노력이 요구된다. 따라서 기능이 교체되는 컴포넌트의 상황과 범위에 적절하게 기능 테스트가 수행되어야 한다. 빈번한 컴포넌트의 교체를 통한 시뮬레이션 환경에서 컴포넌트가 교체되는 시점에 기능 테스트를 수행하고 컴포넌트의 기능이 정확할 때 전체적인 관점에서 성능을 판별하는 방법이 타당하고 합리적인 시뮬레이션 방법이 될 것이다.

그림 3은 본 논문에서 제안하는 교체 컴포넌트의 기능 테스트를 위한 테스트 프레임워크의 모습이다. 이 프레임워크는 기본적으로 컴포넌트의 단위 테스트를 수행하기 위한 목적이지만 컴포넌트들 간의 통합 및 연동 테스트를 수행하기 위한 테스트 프레임워크로도 활용할 수 있다.



(그림 3) 컴포넌트 기능 테스트 프레임워크

테스트 프레임워크의 구성 요소는 다음과 같다. 수행 방식은 4 절에서 기술한다.

- 테스트 드라이버: 테스트를 수행하는 수행기
- 교체 컴포넌트: 기존 컴포넌트를 대체하게 될 새롭게 구현된 컴포넌트
- 테스트 결과분석기: 컴포넌트 수행 후 발생한 실행 결과와 저장된 기존 컴포넌트의 실행 결과를 비교하여 테스트의 성공 여부를 판단함
- 컴포넌트 관리자: 컴포넌트를 컨테이너에 포함시킬 때 컴포넌트의 등록, 관리 등의 역할

수행, 테스트 진행 시 테스트 대상 컴포넌트에 대한 정보를 제공

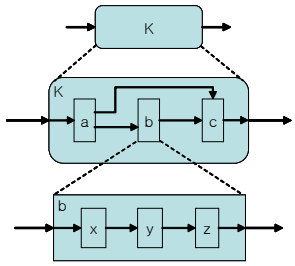
- 테스트 관리자: 테스트 대상 컴포넌트에 적용할 입력 데이터를 선택, 테스트 결과 분석을 위해 테스트 대상 컴포넌트에 대응되는 기존 컴포넌트의 실행 결과를 선택, 테스트 수행 후 테스트 수행 결과 유지

3.4 테스트 케이스 획득

3.4.1 단계적 세분화(stepwise refinement) 및 로깅(logging)을 통한 테스트 케이스 획득

신호생성기의 경우 시뮬레이션 작업의 수행 시간(시작시간~종료시간)과 위성의 초기 위치정보(위성의 좌표정보)가 입력 데이터로 주어지면 시뮬레이션 작업의 수행 결과로 실제 위성이 생성한 신호와 유사한 IF 레벨의 신호 데이터가 생성된다. 신호수신기의 경우 이러한 IF 신호가 입력 데이터로 제공되면, 수행 결과로 사용자의 위치정보를 얻게 된다.

여기서 시뮬레이션 작업의 수행 시간과 위성의 초기 위치정보나 IF 레벨 신호 데이터의 경우 신호생성기와 신호수신기 각각의 입력 데이터이므로 세부 기능의 테스트를 위해서는 그 데이터를 곧바로 사용할 수 없다. 신호수신기의 경우 IF 신호 데이터는 여러 컴포넌트를 거치면서 다양한 형태의 데이터로 세분화 되는데 테스트 대상 컴포넌트가 수신기에서 어느 기능 단위에 위치하느냐에 따라 입력 데이터의 형태는 매우 달라지게 된다. 따라서 각 세부 기능 컴포넌트에 대한 기능 테스트를 수행하기 위해서는 세부 기능들에 적절한 입력 데이터가 준비되어야 한다. 이렇게 개별 컴포넌트마다 다른 입력 데이터를 얻기 위해서는 단계별 세분화 방법을 적용할 수 있다. 단계별 세분화는 그림 4와 같이 최외곽 컴포넌트에서 내부 컴포넌트로 점진적으로 기능을 세분화 해가는 방법이다. 이 방법을 이용하면 특정 컴포넌트의 입력 데이터와 그 컴포넌트 이전에 위치한 컴포넌트 혹은 그 컴포넌트를 포함하는 외곽 컴포넌트의 입력 데이터와의 상관관계를 파악할 수 있고 이를 통해 특정 컴포넌트에 적합한 입력 데이터를 준비할 수 있다.



(그림 4) 모듈의 단계적 세분화 관계

그러나 특정 컴포넌트에 대한 입력 데이터를 매번 단계별 세분화를 통해 파악하는 것은 어느 정도의 노력을 요구한다. 이를 간편하게 해결하기 위한 방법으로 각 기능 단위마다 적절한 로그(log) 기능을 추가하여 입력 데이터와 출력 데이터를 기록할 수 있다. 그림 2의 검증 환경에서 검증 모드의 수행이 끝나고 나면 시뮬레이션 모드에서 시뮬레이션 작업이 진행된다. 실제로 시뮬레이션 작업은 다양한 목적에 따라 여러 컴포넌트들을 교체해 가면서 진행된다. 그러나 본격적인 시뮬레이션 작업에 앞서 시뮬레이션 모드에서 내부 컴포넌트의 교체 없이 여러 번에 걸쳐 시뮬레이션을 수행할 때 각 세부 컴포넌트의 입력 및 출력 위치에 로그 기능을 추가하여 각 컴포넌트에 대한 입력 데이터와 출력 데이터를 수집할 수 있다. 이 작업은 세부 기능 모듈에 대한 컴포넌트 교체 시 기능 테스트를 위해 필요한 테스트 케이스와 테스트 오라클을 수집하기 위한 것이다. 수집된 입력 데이터는 단계별로 세분화되어 표 1과 같은 형태로 저장된다.

(표 1) 입력 데이터의 계층적 세분화

1_{st} level	2_{nd} level	3_{rd} level	4_{th} level
I_k	I_a		
	I_b	I_x	
		I_y	
		I_z	
	I_c		

모듈 y에 대한 입력 데이터를 접근하기 위한 인덱스는 <Iki, Ibi, Iyi>로 표현된다. 여기서 i는 i 번째 시행 과정에서 수집된 데이터를 의미한다. 예를 들어, 그림 4에서 컴포넌트 y를 새로운 컴포넌트 y'으로 교체하려는 상황을 고려하자. y'을 테스트하기 위한 입력 데이터는 표 1에서 인덱스 <Ik, Ib, Iy>로 접근하여 획득할 수 있다.

3.4.2 테스트 케이스의 추가와 윈도우 확장

충분한 테스트를 위해서는 보다 많은 테스트 케이스를 만들 필요가 있다. 이 때 임의로(random) 테스트 케이스를 생성하기 보다는 그림 2의 검증 환경에서 시뮬레이션 모드 수행 시 수집된 입력 데이터를 참조하여 각 입력 파라미터 별로 새로운 테스트 데이터를 만들어 표 1에 추가할 수 있다. 만일 입력 파라미터의 타입이 정수형(int), 문자열(string) 등과 같이 단순한 형태라면 테스트 데이터를 생성하기가 어렵지 않다. 그러나 만일 입력 데이터가 어떤 형태의 전파신호라면 단순한 방법으로 (즉 별도의 체계적인 로직이나 장치 없이) 그런 신호를 생성하기는 어렵다. 후자의 경우 컴포넌트 x가 어떤 장치나 로직을 통해 전파신호를 생성한다면 반드시 컴포넌트 y의 선행 컴포넌트 x의 수행 결과를 새로운 테스트 데이터로 추가하여야 한다. 이와 같이 특정 컴포넌트의 테스트 데이터를 생성하여 추가하고자 할 때, 해당 컴포넌트의 인터페이스 정보와 입력 로그만으로 테스트 데이터를 생성하기 어려울 경우에는 선행 컴포넌트를 함께 고려하여 테스트 데이터를 생성할 수 있다. 이런 상황을 윈도우 확장(window widening)이라 정의한다. 이렇게 불리는 이유는 관심 대상이 컴포넌트 y에서 컴포넌트 x와 y로 넓어졌기 때문이다. 이 경우에 컴포넌트 x의 수행 결과를 파일에 저장할 수 있으면 저장된 수행 결과를 테스트 데이터로 추가하면 된다. 그러나 만일 x의 수행 결과가 저장하기 어려운 형태라면 y를 테스트하기 위해서는 컴포넌트 x의 입력을 가지고 x를 수행하고 그 결과가 곧바로 컴포넌트 y에 전달될 수 있도록 테스트가 수행되어야 한다.

이런 상황을 수용할 수 있도록 테스트 프레임워크는 설계되었다. 그림 3의 테스트 컨테이너 내부에서 교체 컴포넌트 앞에 위치한 컴포넌트 D는 교체 컴포넌트가 단독으로 테스트될 수 있을 때는 제거된다. 그러나 윈도우 확장 상황에서 선행 컴포넌트의 수행 결과를 저장하기 곤란한 경우에는 테스트 대상 컴포넌트의 선행 컴포넌트로 대체된다. 즉 컴포넌트 x → y 순서로 수행되던 상황이 x → y'의 순서로 수행될 수 있도록 함으로써 교체 컴포넌트인 y'의 기능을 테스트 할 수 있다.

3.5 테스트 오라클 획득

3.5.1 단계적 세분화 및 로깅을 통한 테스트 오라클 획득

테스트 오라클(Test Oracle)은 테스트 수행의 결과가 성공인지 혹은 실패인지를 판단하기 위한 데이터나 판단 메커니즘을 말한다[10].

위성항법 시뮬레이션의 경우, 동일 기능의 여러 컴포넌트들을 교체해 가면서 기능이나 성능적인 측면에서 시뮬레이션하고자 하므로 동일 기능을 구현하고 있는 컴포넌트는 기존 컴포넌트의 수행 결과에 부합하여야 한다. 따라서 기존 컴포넌트의 수행 결과를 교체 컴포넌트에 대한 테스트 오라클로 활용할 수 있다.

테스트 데이터 생성 방법과 마찬가지로 단계적 세분화를 통해 테스트 오라클도 각 컴포넌트에 맞게 세분화 할 수 있다. 또한 입력 데이터와 마찬가지로 로그 기능을 통해 수집된 출력 데이터도 단계적으로 세분화되어 출력 데이터 표에 저장된다. 이 때 모듈 y에 대한 출력 결과의 인덱스는 <Oki, Obi, Oyi>로 나타낼 수 있다.

예를 들어, 그림 4에서 컴포넌트 y를 새로운 컴포넌트 y'으로 교체하려는 상황을 고려하자. y의 수행 결과가 y'에 대한 테스트 오라클의 역할을

할 때 이것이 비교하기 쉬운 형태라면 이것을 테스트 오라클로 활용한다. 이 테스트 오라클은 출력 데이터 표 2에서 인덱스 <Ok, Ob, Oy>로 접근하여 획득할 수 있다.

(표 2) 출력 데이터의 계층적 세분화

1st level	2nd level	3rd level	4th level
O_k	O_a		
	O_b	O_x	
		O_y	
		O_z	
	O_c		

3.5.2 윈도우 확장을 통한 테스트 오라클 획득

만일 특정 컴포넌트의 출력 데이터가 전파신호와 같은 형태라면 해당 컴포넌트와 교체 컴포넌트의 출력 데이터를 맞비교하기란 쉽지 않다. 이런 상황에서는 특정 컴포넌트의 결과 데이터를 입력으로 사용하는 후행 컴포넌트까지 범위를 넓혀 후행 컴포넌트의 수행 결과를 포함하여 테스트 오라클로 사용한다. 이런 과정을 테스트 오라클을 위한 윈도우 확장이라 정의한다. 즉 컴포넌트 y와 y'의 결과 데이터를 맞비교하는 대신에 컴포넌트 z의 수행을 포함하여 y → z의 수행 결과와 y' → z의 수행 결과를 비교한다.

테스트 프레임워크는 이런 상황도 수용할 수 있다. 그림 3의 테스트 컨테이너 내부에서 교체 컴포넌트 뒤에 위치한 컴포넌트 D는 교체 컴포넌트만으로 기능의 정확성 여부를 판단할 수 있을 때는 제거되지만, 테스트 오라클에 대한 윈도우 확장이 요구되는 상황에는 테스트 대상 컴포넌트의 후행 컴포넌트로 대체된다. 즉 먼저 컴포넌트 y → z 순서로 수행하고, 교체 컴포넌트 y'에 대해 y' → z의 순서로 수행함으로써 교체 컴포넌트 y'의 기능을 테스트 할 수 있다.

3.5.3 테스트 오라클에서의 오차

컴포넌트의 수행 결과 데이터로부터 테스트 오라클을 생성하는 과정에서는 오차를 고려하여야 한다. 사실 기존 컴포넌트의 수행 결과와 교체 컴포넌트의 수행 결과가 완전하게 일치하지 않을 수 있다. 예를 들어 신호수신기에서의 위치 정확도 향상을 위한 개선된 알고리즘을 적용할 경우 결과적으로 생성되는 위치 정보는 기존의 테스트 오라클보다 더 정밀한 위치를 산출하기 때문에 그 결과가 정확히 일치하지 않을 수 있다. 또한 신호생성기에서 생성한 신호의 경우 최종 생성된 결과가 항상 같지는 않다. 이는 실제 신호와 유사한 신호를 생성하기 위해 대류층, 이온층과 같은 특성을 반영하여 생성하기 때문이다. 이외에도 다양한 불일치는 약간의 오차 때문에 발생한다. 따라서 생성될 테스트 오라클은 기존 컴포넌트의 수행 결과에 허용 오차 범위를 반영하여 설계되어야 한다. 최종적으로 테스트 결과 분석기에서는 다음과 같은 기준을 적용하여 수행된 테스트의 결과를 판정한다.

[테스트 성공 조건]

```
if ¦Rnew - Roriginal¦<= Eδ, then
        success
else
        fail
```

여기서, Roriginal은 테스트 오라클로 사용될 기존 컴포넌트의 수행 결과이고, Rnew는 교체된 컴포넌트의 수행 결과이다. Eδ는 허용 오차를 의미한다.

3.6 테스트 충분성 평가

테스트를 진행할 때 테스트 수행을 어느 정도 반복해야 하는지에 대한 기준이 필요하다. 테스트 충분성(Test Adequacy)은 그런 기준을 제시한다[11].

그림 2의 검증 환경에서 검증 모드의 수행이 끝나면 테스트 케이스와 테스트 오라클을 수집하기 위한 목적으로 시뮬레이션 모드의 작업이 진행된다. 이 때에 진행될 시뮬레이션의 반복 횟수

는 일정하게 정하는 것이 좋다. 정해진 수행 횟수를 통해 수집된 테스트 케이스와 테스트 오라클은 세부 기능 모듈에 대한 테스트 충분성을 정의할 때 사용된다. 시뮬레이션 모드에서 반복 작업의 수행 횟수는 다음과 같이 정의한다.

[시뮬레이션 모드에서 반복 작업 수행 횟수]

Πi=1 k n(pi)

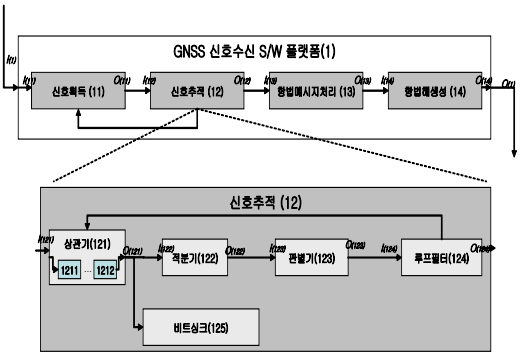
여기서, n(pi)는 소프트웨어 신호생성기(Ssw)의 입력 파라미터 pi의 입력 도메인에서 동등 클래스의 수를 의미한다.

예를 들어, Ssw의 입력 파라미터는 시뮬레이션 작업 시간, 위성의 초기 위치정보이다. 시뮬레이션 작업 시간을 위성의 위치 정보가 갱신되는 두 시간 단위로 나누어 12개의 동등 클래스로 구분하고, 위성의 위치 정보를 가시 범위 내와 가시 범위 밖으로 2개의 동등 클래스로 나누어진다고 할 때, 전체 24개의 입력 조합이 생긴다. 따라서 시뮬레이션 모드는 24번 시행되고 이 때 생성된 입력 데이터와 수행 결과를 기능 테스트를 위한 기본 테스트 케이스와 테스트 오라클로 삼는다.

앞서 언급한 바와 같이 이러한 테스트 케이스와 테스트 오라클은 세부 모듈의 기능 테스트 과정에서 단계적으로 세분화되어 적용된다. 따라서 특정 컴포넌트의 테스트에 필요한 테스트 케이스는 세분화 과정에서 결정된다. 특정 컴포넌트의 기능 테스트에서 테스트 충분성 여부는 세분화 과정에서 준비되었던 테스트 케이스가 모두 적용되었을 경우 테스트가 충분히 되었다고 판단한다. 예를 들어, 위와 같이 24개의 기본 테스트 케이스가 준비되었다면, 이 24개의 테스트 케이스를 수행하여야만 교체 컴포넌트의 테스팅 충분히 되었다고 판단한다.

4. 시뮬레이션 테스트 프레임워크 적용

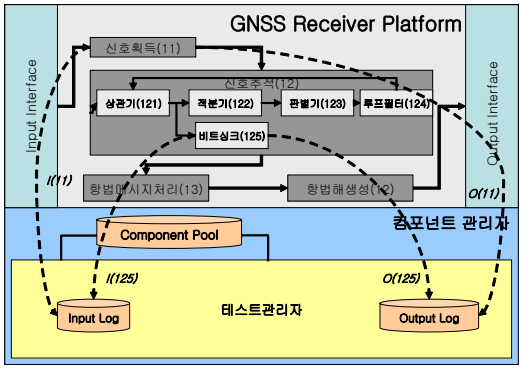
본 논문의 테스트 프레임워크는 현재 한국전자통신연구원에서 개발중인 소프트웨어 기반 위성항법 시뮬레이터[3]에 적용되고 있다. 그림 1의 위성항법 시뮬레이터 중 위성항법 신호수신 플랫폼은 그림 5와 같이 단계적으로 세분화된다.



(그림 5) 위성항법 신호수신 플랫폼 단계적 세분화

그림 5에서 ()안의 숫자는 각 컴포넌트를 구분하기 위한 구분자(id)이다. 우선 위성항법 신호수신 플랫폼 자체는 내부적으로 신호획득, 신호추적, 항법메시지처리, 항법해(측위) 생성 컴포넌트로 구성된다. (12)의 신호추적 컴포넌트는 다시 내부적으로 상관기, 적분기, 판별기, 루프필터, 비트싱크 컴포넌트로 구성되며, 각 컴포넌트는 다시 하위의 알고리즘을 구현한 컴포넌트들로 구성된다. 이러한 단계적 세분화는 교체 가능한 최하위의 알고리즘 컴포넌트까지 세분화된다.

위성항법 신호수신 플랫폼에서 테스트 케이스 및 테스트 오라클의 생성은 앞서 설명한 것과 같이 시뮬레이션 모드에서 기록되는 각 컴포넌트의 입/출력 데이터를 이용한다. 그림 6은 위성항법 신호수신 플랫폼의 시뮬레이션 모드에서 테스트 케이스 및 테스트 오라클을 저장하는 과정을 보여준다. 위성항법 신호수신 플랫폼이 시뮬레이션 모드에서 동작하는 동안 각 내부 컴포넌트는 입/출력 데이터를 로그로 기록한다.



(그림 6) 위성항법 신호수신 플랫폼에서 테스트 케이스 / 테스트 오라클 획득

이러한 데이터들은 테스트 프레임워크에 입력 데이터 로그, 출력 데이터 로그에 기록되어 테스트에 활용되게 된다. 다음 표 3 과 4는 위성항법 수신 플랫폼의 테스트 케이스와 테스트 오라클의 목록이다.

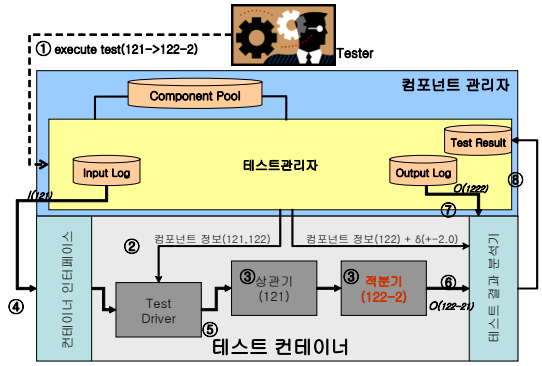
(표 3) 위성항법 신호수신 플랫폼 테스트 케이스

테스트 케이스 (I_{11})	테스트 케이스 (I_{12})	테스트 케이스 (I_{121})	테스트 케이스 (I_{122})
IF Data	IF Data	IF Data	I(E/P/L)AccCorrValue
	PRN	PRN	Q(E/P/L)AccCorrValue
	Doppler	Doppler	
	Code Phase	Code Phase	
	Acquisition S/F	Acquisition S/F	
Configuration Values	*Configuration Values*	*Configuration Values*	*Configuration Values*

(표 4) 위성항법 신호수신 플랫폼 테스트 오라클

테스트 오라클 (O_{11})	테스트 오라클 (O_{12})	테스트 오라클 (O_{121})	테스트 오라클 (O_{122})
PRN	Data Bit	I(E/P/L)AccCorrValue	I(E/P/L)IntCorrValue
Doppler	I(E/P/L)	Q(E/P/L)AccCorrValue	Q(E/P/L)IntCorrValue
Code Phase	Q(E/P/L)		
Acquisition S/F	Frequency Error		
	Code Error		
	Frequency ErrGroup		
	Code ErrGroup		

위의 테스트 케이스, 테스트 오라클은 단계적으로 세분화된 하위 컴포넌트에 맞도록 단계적으로 세분화되어 저장된다. 이러한 데이터를 이용해 적분기(122) 컴포넌트의 교체에 대한 테스트 수행 환경은 다음과 같다.



(그림 7) 적분기 컴포넌트 교체 테스트

테스트 수행은 다음과 같은 순서로 이루어진다. 우선 테스터는 테스트 관리자에 테스트를 수행할 컴포넌트를 지정한다. 필요한 경우 윈도우 확장(window widening)을 위해 테스트 수행에 필요한 컴포넌트의 목록을 함께 입력한다. 테스트 관리자는 테스트 컨테이너에 테스트 대상 컴포넌트에 대한 정보를 전달하고, 테스트 컨테이너는 컴포넌트 풀에서 대상 컴포넌트를 로드(load)한다. 테스트 준비가 완료되면 테스트 관리자는 입력 데이터 로그에서 해당 컴포넌트의 테스트 케이스를 찾아 컨테이너 인터페이스에 전달하고, 테스트 드라이버는 이를 받아 테스트 대상 컴포넌트를 구동시킨다. 테스트 수행결과는 테스트 결과 분석기에 전달되며, 테스트 결과 분석기는 출력 데이터 로그를 검색하여 테스트 오라클로 받는다. 테스트 결과와 테스트 오라클을 서로 비교하여 테스트 수행 결과를 판단하고 이를 테스트 수행 결과로 기록한 다음 테스트 수행을 완료한다.

5. 결 론

시뮬레이션은 컴퓨터를 이용하여 실제 사물이나 작업의 상태, 혹은 프로세스를 모방하여 그 특징을 찾아내는 작업을 지칭하며, 시뮬레이터는 이러한 시뮬레이션 작업을 수행하는 도구를 말한다. 이러한 시뮬레이션 작업의 특성상 다양한 시뮬레이션 알고리즘의 교체 및 변경이 빈번하게 발생하며, 이에 대한 적절한 기능 테스트가 수행되어야 한다. 그러나 기존의 시뮬레이션 시스템은 특정 알고리즘의 교체 후 시스템 수준에서 기능 테스트를 수행 함으로 인해 시뮬레이션 작업의 효율을 심각하게 떨어뜨렸다. 따라서 시뮬레이션 알고리즘을 구현한 컴포넌트 레벨의 기능 테스트를 적용할 수 있는 방안이 요구되었다. 특히 위성항법 시뮬레이터는 향후 개발될 갈릴레오 및 현대화된 GPS 위성을 겨냥해 다양한 시뮬레이션 알고리즘의 개발이 이루어지고 있기 때문에 이를 효과적으로 지원하기 위한 적절한 테스트 방법 및 프레임워크가 필요하다. 이에 본 논문에서는 위성항법 시뮬레이션 작업을 동적으로 지원하는 테스트 프레임워크를 제안하였다. 본 논문의 테스트 기법은 시뮬레이션 시스템의 세부 컴포넌트의 단계적 세분화를 통해 테스트 케이스와 테스트 오라클을 추출하고, 이를 기반으로 교체되는 컴포넌트를 테스트 할 수 있는 환경을 제공한다. 따라서 현재 개발중인 알고리즘뿐만 아니라 향후 개발될 알고리즘의 검증에도 널리 활용될 수 있을 것으로 판단된다.

향후 본 논문의 테스트 방법을 좀 더 일반화하여 다양한 시뮬레이터 시스템에 적용 가능한 컴포넌트 테스트 방법을 구축할 계획이며, 기능 테스트뿐만 아니라 성능 테스트에 활용하는 방법에 대해서도 연구를 수행할 예정이다.

참 고 문 헌

[1] C.A. Chung, SIMULATION MODELING HANDBOOK: A Practical Approach, CRC Press, 2004.

[2] A. Sulistio, C.S. Yeo, and R. Buyya, “Taxonomy of computer-based simulation and its mapping to parallel and distributed system simulation tools,” Software Practice and Experience, pp653-673, 2004.

[3] 이재은, 주인원, 이상욱, 김재훈, “소프트웨어 위성항법 시뮬레이터 예비 설계”, 한국항공우주학회 학술발표대회 논문집, pp. 395~398, 2008.

[4] S. Kang, “Relating Interoperability Testing with Conformance Testing,” Global Telecommunications Conf. (GLOBECOM 98), Vol. 6, pp.3768-3773, 1998.

[5] J. Gao, D. Gopinathan, Q. Mai, and J. He, “A Systematic Regression Testing Method and Tool for Software Components,” 30th Annual Int’l Computer Software and Applications Conf. (COMPSAC’06), pp. 455-466, 2006.

[6] J. Bach, “General Functionality and Stability Test Procedure,” http://www.satisfice.com/tools/procedure.pdf, 1999.

[7] 오일노, 국승학, 김현수, 김철홍, 윤석진, 최유희 “J2ME 대응 라이브러리 클래스 테스트 도구 개발”, 한국소프트웨어 공학 학술대회 논문집, 제8권, pp 357-364, 2006.

[8] T. Bennett and P. Wennberg, “Maintaining Verification Test Consistency between Executable Specifications and Embedded Software in a Virtual System Integration Laboratory Environment,” Proc. of the 28th Annual NASA Goddard Software Engineering Workshop, pp.221-228, 2003.

[9] http://www.mathworks.co.kr/products/simverification/

[10] M. L. Hutcheson, Software Testing Fundamentals: Methods and Metrics, Wiley & Sons, 2003.

[11] A. P. Mathur, Foundations of Software Testing, Pearson Education, 2008.

◐ 저 자 소 개 ◑

국 승 학

2004년 충남대학교 컴퓨터과학과 졸업(학사)
2006년 충남대학교 컴퓨터공학과(공학석사)
2006년 - 현재 충남대학교 컴퓨터공학과 박사 재학
관심분야 : 소프트웨어 테스팅, 소프트웨어 품질관리, SOA, 웹 서비스
e-mail : triple888@cnu.ac.kr

김 현 수

1988 서울대학교 계산통계학과 졸업
1991 한국과학기술원 전산학과(공학석사)
1995 한국과학기술원 전산학과(공학박사)
1995 - 1995 한국전자통신연구원 Post Doc
1996 - 1998 금오공과대학교 전임강사
1998 - 2001 금오공과대학교 조교수
1999 - 2000 Colorado State University 방문교수
2001 - 2002 충남대학교 공과대학전기정보통신공학부 컴퓨터전공 조교수
2002 - 2007 충남대학교 공과대학 전기정보통신공학부 컴퓨터전공 부교수
2007 - 2008 Purdue University 방문교수
2007 - 현재 충남대학교 공과대학 전기정보통신공학부 컴퓨터전공 교수
관심분야 : 소프트웨어 테스팅, 소프트웨어 재공학, 컴포넌트 마이닝 컴포넌트 테스팅, SOA
e-mail : hskim401@cnu.ac.kr

이 상 욱

1988년 2월 : 연세대학교 천문기상학(이학사)
1991년 3월 : 미국 Auburn대학교 항공우주공학(석사)
1994년 3월 : 미국 Auburn대학교 항공우주공학(박사)
1993년 3월 ~ 현재 : 한국전자통신연구원, 책임연구원
관심분야 : 위성제어 및 관제, 위성항법 및 항법 응용
e-mail : slee@etri.re.kr