한국시뮬레이션학회 논문지
Vol. 19, No. 1, pp. 33-39 (2010. 3)

이벤트 기반 웹서비스를 이용한 워게임 시뮬레이터 제작

이재민[1] · 김병철[1] · 김태섭[1] · 이강선[1†]

War-game Simulator Using Event based Web Services

Byoung Chul Kim · Jae Min Lee · Tae Sup Kim · Kang Sun Lee

ABSTRACT

As future warfare becomes network-centric, war-game simulators require high interoperability between networked forces and dynamic reconfiguration in accordance with war events. In this paper, we propose an event-driven methodology to develop dynamic war-game simulations. Federates are developed by event-driven web services. The event-driven web services consistently sense war events and response them only if they are interested. By the sense-and-response mechanism and asynchronous event processing, we are able to save simulation time. An Anti-Surface-Warfare simulator is constructed to demonstrate the methodology and suggests that event-driven web services are efficient to model and simulate warfare where numerous events are generated from hardware systems and people dispersed on the network.

Key words : Defense modeling and simulation, Event-driven web service, Dynamic composition

요 약

미래 전장환경이 네트워크 중심으로 변해감에 따라, 워게임 시뮬레이터는 네트워크에 분산된 모듈간의 높은 상호운영성과 전장 이벤트에 따른 동적구성의 필요성이 높아지고 있다. 본 논문에서는 이벤트 기반의 워게임 시뮬레이터 개발방법론을 제안한다. 워게임 시뮬레이터의 페더레이트들은 이벤트 기반의 웹서비스로 개발되며, 각 페더레이트는 전장 이벤트를 감지하고, 관계있는 이벤트가 발생할 경우에만 워게임에 반응하게 된다. 이러한 감지-반응 방법과 비동기적 이벤트 처리방법을 이용하여 시뮬레이션 수행 시간을 줄일 수 있다. 본 논문에서는 간단한 수상전 시뮬레이터를 구성하여, 제안된 방법이 전장 장비 및 네트워크상에서 모델러 및 운영자를 통해 발생되는 다양한 이벤트를 처리해야 하는 미래 전장환경 시뮬레이션 수행시 효과적임을 보이도록 한다.

주요어 : 국방 모델링 및 시뮬레이션, 동적 웹서비스 조합, 이벤트 기반 웹서비스

1. 서 론

미래 전장이 네트워크 중심전의 양상으로 진화됨에 따라[1], 워게임 시뮬레이터는 네트워크상에 분산된 각종 페더레이트 및 자원들 간의 효율적인 상호 운영성을 지원할 필요가 증대되고 있다. 기존 HLA(High-Level Architecture) 기반 페더레이트[2]는 국방자원간의 효율적인 상호 운영성을 지원하는데는 효과적이나, 일반 엔터프라이즈 영역의 자원 및 페더레이트와의 상호 운영에는 제한이 있다[3]. 최근 등장한 HLA+SOA(Service Oriented Architecture)는 이러한 HLA의 제한점을 인식하여 엔터프라이즈 영역의 여러 자원을 국방 모델링 및 시뮬레이션에서 결합하려는 시도로, 네트워크 중심전을 모의 실험하기 위한 효율적인 방안으로 검토되고 있다[4,5].

웹서비스는 HLA+SOA 프레임워크를 실현시키는 핵심기술로, 표준 기술방식 및 표준 웹 프로토콜을 통해 프로그래밍 언어 및 플랫폼에 독립적으로 재사용 가능한 시뮬레이션 모델을 구축할 수 있도록 한다. 그러나 일반적인 웹서비스는 요청/응답방식으로, 미리 정의된 일련의 순차적 프로세스를 반복적으로 수행하는 경우에는 적합

* 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(UD080042AD)
2009년 7월 1일 접수, 2010년 2월 17일 채택
[1] 명지대학교 컴퓨터공학과
주 저 자 : 이재민
교신저자 : 이강선
E-mail; ksl@mju.ac.kr

하나, 워게임 시뮬레이션과 같이 전장에서 발생하는 다양한 이벤트에 따라 상이한 시뮬레이션 과정을 수행하게 되는 경우에는 효율적이지 못하다[6]. 참고문헌[7]에 따르면 일반적인 웹서비스 기반 페더레이트는 HLA/RTI 기반 페더레이트에 비해 수행시간이 오래 걸리는 것으로 조사된 바 있다.

본 연구에서는 이벤트 기반 웹서비스를 이용한 워게임 시뮬레이션 제작 방법론을 소개하여, 국방 및 일반 엔터프라이즈 영역의 재사용 자원들간에 상호 운영성을 높이고 일반적인 웹서비스 기반 시뮬레이터에 비해 수행 속도를 개선할 수 있는 기술을 제시하도록 한다.

본 논문에서 제안한 방법에서는 전장환경에서 발생되는 각종 이벤트를 규격화하여 표현한 후, 웹기반 페더레이트가 관심 있는 이벤트를 등록하도록 한다. 이후, 각 웹기반 페더레이트는 해당 이벤트가 발생할 경우에만 주어진 임무를 수행 하여 시뮬레이션의 효율성을 고려하도록 하였다. 또한, 기존 웹기반 시뮬레이션에서는 현재 이벤트에 대한 처리가 완료되기 전까지는 새로운 이벤트의 입력을 차단하는 방식과는 달리, 본 연구에서는 비동기적으로 이벤트를 처리하는 방식을 지원함으로써 효율성을 더욱 증대시킬 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구 및 이벤트 기반 웹서비스 기술을 소개 하고, 3장에서는 이벤트 기반 웹서비스 기술을 이용하여 워게임 시뮬레이션을 제작하는 방법을 소개한다. 4장에서는 제안된 방법론에 따라 수상전 시뮬레이터를 작성한 후 시뮬레이션 수행 시간의 개선 정도를 실험을 통해 제시하도록 한다. 5장에서는 연구 결과를 평가하고 향후 연구를 제시한다.

2. 이벤트 기반 웹서비스 기술

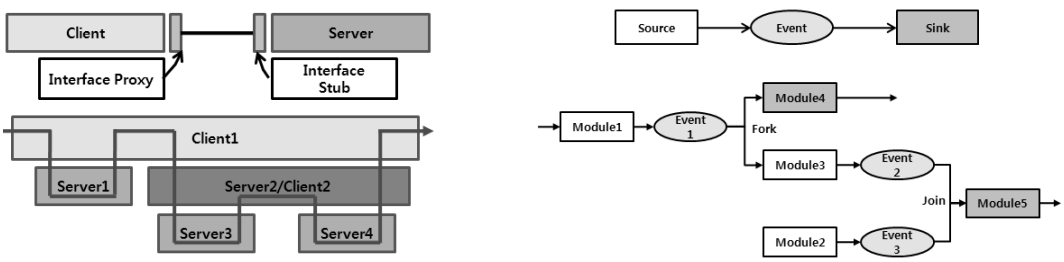
웹서비스의 구성방식에는 서비스 기반 구조(SOA)와 이벤트 기반 구조 (EDA: Event Driven Architecture)가 있다. SOA는 요청에 응답하여 주어진 서비스를 제공하는 것을 기본으로 하는 구성방식 이고, EDA는 이벤트를 발생 시키고 발생된 이벤트를 처리하는 각 모듈들이 연합해서 수행되는 구성방식이다.

그림 1(a)와 같이 SOA는 Interface Proxy와 Interface Stub을 이용하여 클라이언트의 요청에 대한 서버의 응답으로 수행된다. 클라이언트가 흐름을 제어하며 요청에 대한 응답이 오기 전까지 다른 입력(Request)을 처리할 수 없다. 반면, 그림 1 (b)와 같이 EDA는 이벤트가 처리되는 모듈(Consumer)이 흐름을 제어하게 되고, 이벤트의 처리 결과에 따라 여러 모듈에게 처리를 분기(Fork)하거나 여러 개의 처리결과를 종합(Join)하여 처리하는 등 병렬 처리가 가능하다. 이와 같이 EDA는 “이벤트 발생-이벤트 처리”의 단순 반복 구조에서 다른 입력(Event)를 같이 처리할 수 있다. 다음 표 1은 SOA와 EDA를 비교 요약한 것이다.

일반적인 SOA방식의 웹서비스를 이용하여 시뮬레이터를 작성한 대표적인 연구로는 HLA Web Service API 방식과 페더레이트 브릿지 방식이 있다[3,5]. HLA Web Service API 방식은 그림 2와 같이 서버에 WSPRC를 두고 여러 웹서비스 페더레이트를 같은 WSPRC에 연결한다.

표 1. SOA와 EDA

구분	SOA	EDA
상호 규약 정보	서비스 인터페이스 정보	이벤트 규격 정보
연결 방식	1 : 1	N : N
흐름제어 주체	클라이언트	이벤트 수신자
흐름제어 방식	순차경로	동적/병렬/비동기 방식
새로운 입력에 대한 대응	진행중 차단	진행중 반응



(a) SOA 개념도 (b) EDA 개념도

그림 1. 시뮬레이터 개발 환경의 아키텍처

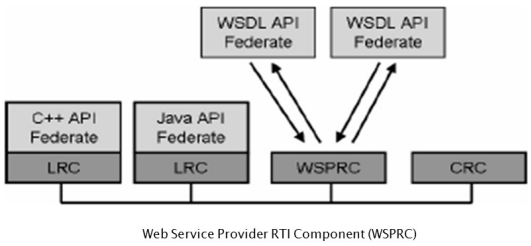


그림 2. 웹서비스 API를 포함한 HLA 구조도

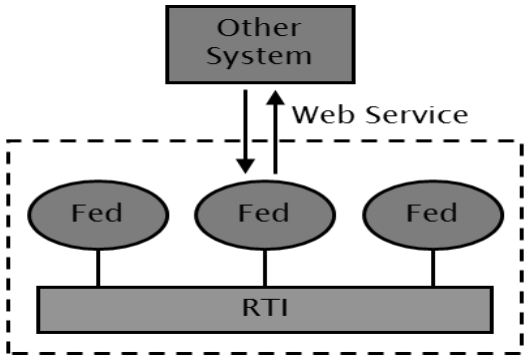


그림 3. 하나의 페더레이트를 브릿지로 사용

페더레이션 브릿지방식은 그림 3과 같이 페더레이션들 사이에 "브릿지"의 역할을 하는 페더레이트를 사용한다[5]. 페더레이트 브릿지 방식은 HLA/RTI 기반 페더레이트와 웹서비스 페더레이트를 연결 해주며, 이를 위해 웹서비스의 상태를 관리한다. 이러한 방식에서는 WSPRC 및 브릿지와 같은 웹서비스 전담 모듈이 모든 페더레이트로부터 발생되는 서비스 요청 및 RTI와의 연결을 담당하게 되며, 각 페더레이트는 서비스 요청 후 응답이 올 때 까지 대기하는 방식으로 작동되어 순수 HLA/RTI 기반 시뮬레이션 보다 속도 저하가 발생된다[7].

이벤트 기반 웹서비스 기술은 이벤트 기반 아키텍처를 웹서비스로 구현한 것으로, 대표적으로 WS-Notification[8]이 있다.

WS-Notification은 그림 4에 도식화된 것과 같이 Topic, Publisher, Broker, Subscriber, Consumer로 구성된다. Publisher, Broker, Subscriber, Consumer의 간략한 설명은 다음과 같다.

- Topic : 이벤트의 유형. 예를 들어 에러메시지, 정보메시지 등 이벤트의 유형을 대표한다

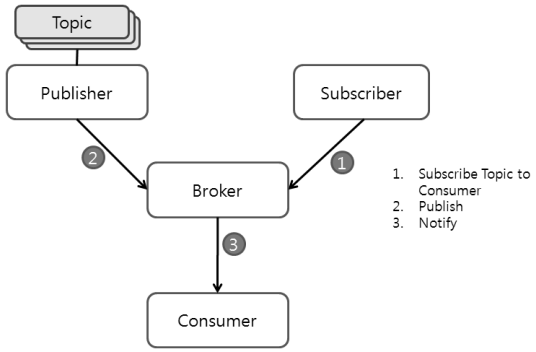


그림 4. WS-Notification

- Publisher : 이벤트의 생산자
- Subscriber : 누가, 어떤 종류의 이벤트를 수신하는지 Broker에 등록하는 등록자
- Consumer : 전달된 이벤트를 처리하는 소비자
- Broker : 생산자에 의해서 생성된 이벤트를 전달 받고, 등록자에 의해 등록된 소비자에게 이벤트를 알려(Notify) 주는 대리자

그림 4는 WS-Notification에 따른 "이벤트 발행-구독" 구조를 나타낸다. Publisher와 Broker에 등록된 Topic에 관한 이벤트가 발생하면, Publisher는 Broker에 이벤트 발생 사실을 알려주고, Broker는 Subscriber가 등록한 Consumer에게 이벤트가 발생한 사실을 통지한다.

이벤트 기반 웹서비스 기술의 적용을 통해 기존의 워게임 시뮬레이션에서 이벤트 발생을 감시하던 부분을 이벤트 처리모듈로 독립시켜 이벤트의 발생을 감시할 때 발생하는 오버헤드를 감소시켜 시스템 자원을 효율적으로 사용할 수 있도록 한다. 제안된 방법은 3장에서 자세히 살펴보도록 한다.

3. 이벤트 기반 웹서비스 기술을 이용한 워게임 시뮬레이션 제작 방법

웹서비스 기반의 시뮬레이터 제작에 이벤트 기술을 적용하기 위해서는 다음과 같은 사항을 고려하여야 한다.

- 이벤트 식별: 동일 Topic의 이벤트가 여러 Model에 등록(Subscribe)되어 있는 경우 발생한 이벤트가 어느 모델에서 처리되어야 할 이벤트인지 식별해야 한다.
- 이벤트 처리 순서: 분산 환경에서 발생된 이벤트의

처리순서를 도착 시간을 기준으로 할 것인지 혹은 발생 시간을 기준으로 할 것인지를 결정해야 한다.
- 이벤트 처리 방법: 이벤트 처리 순서를 보장 하기 위한 체계 적인 방법으로, 예를 들어 이벤트 발생 시간 순 으로 처리할 경우, 먼저 발생된 이벤트가 늦게 도착 하였을 때 처리 방법에 대한 규정을 의미한다.

본 논문에서는 이벤트 식별을 위해 이벤트 메시지에 이벤트 수신자의 이름을 표시하도록 하였으며, 이벤트를 수신한 모든 모델이 이벤트의 수신지가 자기 자신인지 검사하도록 했다. 이벤트 처리 방식은 이벤트 발생 시간 순으로 처리하고, 이벤트 발생순서와 도착순서는 같다고 가정한다.

그림 5는 <이벤트 수신자(Consumer)/이벤트 생성자(Publisher)/이벤트 메시지> 형태로 구성된 이벤트를 식별하고 수행하는 이벤트 수행모듈을 구현한 예제이다. 이벤트를 수신한 페더레이트는 그림 5의 18-21 줄에 나타난 바와 같이, 먼저 해당 이벤트가 자기가 실행하는 이벤트인지 식별한 후(Identify Event Sink), 해당하는 이벤트일 경우에만 이벤트를 수행하게 된다.

4. 예제: 수상전 시뮬레이터

본 장에서는 웹서비스 기반의 "수상전 시뮬레이터" 예제를 통해 이벤트 기술의 적용 이점을 보인다. 본 수상전 시뮬레이터는 아군과 적군의 1:1 교전 상황을 모의하여 작성하였으며, 이벤트처리기를 포함하여 총 4개의 페더레이트로 구성된다.

```
1  public void notify(String message) {
2    String targetId       = "";
3    String sourceId       = "";
4    String msg            = "";
5
6    StringTokenizer       st = new StringTokenizer(
7      message
8      , NotificationMessage.MESSAGE_DELIMITER
9    );
10
11   // Parsing TargetId
12   targetId      = st.nextToken();
13   // Parsing SourceId
14   sourceId      = st.nextToken();
15   // Parsing Message
16   msg                   = st.nextToken();
17
18   // Identify Event Sink
19   if ( targetId.equals(this.id) ){
20     // Process Message
21   }
22 }
```

그림 5. 이벤트 식별 예제 코드

- 아군 페더레이트 : 아군함의 기동 및 탐지를 수행하며, 탐지 및 식별 페더레이트로부터 적군이 발견되었다는 이벤트를 받으면, 이벤트 정보(적과의 거리, 적 종류 등)를 기반으로 현 상황에서 가장 적합한 적에 대응할 무기를 선택 한다. 이후 적과 교전 및 분석을 수행한다.
- 적군 페더레이트 : 적함의 기동, 탐지, 교전, 분석을 수행하며, 아군 페더레이트와 같은 논리 아래 수행된다.
- 모니터링 페더레이트 : 아군 및 적군의 페더레이트를 모니터링하고 시뮬레이션 진행 상황을 모니터링한다.
- 이벤트 처리 페더레이트 : 아군 및 적군 페더레이트로부터 이벤트를 수신하며, 수신한 이벤트를 적절한 페더레이트로 전달한다.

페더레이트들은 웹서비스로 제작되어있으며 각 페더레이트의 자세한 설명은 참고문헌[9]에 소개 되어 있다. 본 예제에서는 수상전을 모의하기 위해 기동(Movement), 탐지 및 식별 (Detection & Identification), 교전 (Engagement), 결과 분석 (Analysis of Results)에 필요한 이벤트를 고려하였다. 예를 들어 탐지 및 식별을 위한 *적탐지, 적종류 식별*, 교전을 위한 *무기 할당, 무기발사* 등의 다양한 이벤트를 고려하였다.

그림 6은 "기동" 이벤트를 처리하는 과정을 나타낸다. 아군 페더레이트가 "기동" 이벤트를 발생시키면 이벤트 처리기가 이벤트를 수신하여 이벤트 Topic에 등록되어 있는 모든 이벤트 수신자에게 이벤트의 발생 사실을 통지(Notify)한다. 각 이벤트 수신자는 수신된 이벤트가 자기가 수행해야 할 이벤트인지 이벤트 식별과정을 거친다. 그림 6의 적군 페더레이트의 경우 아군 페더레이트가 생성한 "기동" 이벤트의 수신자(Event Sink)이므로 "기동" 이벤트에 대한 대응을 수행한다. 여기서는 수상전 시나리오에 따라 "적 탐지"이벤트의 발생확률을 계산한다. 이후 "적 탐지" 이벤트의 처리도 "기동" 이벤트와 유사하게 진행된다.

본 논문에서는 웹서비스들이 순차적으로 조합되는 기존의 SOA 방식의 조합과의 비교를 통해 이벤트 기반의 웹서비스 조합 방식이 웹서비스의 성능, 특히 CPU자원의 효율적인 사용면에서 어떠한 이점이 있는지 실험을 통해 확인하였다.

그림 7는 수상전 시뮬레이터를 SOA방식과 EDA방식으로 구현하여 동일한 교전논리와 시뮬레이션 시나리오를 적용하여 반복 실험한 실험 결과를 도식화 한 것이다. 그림 7(a)에서 A, B, C, D는 각각 "기동", "탐지", "교

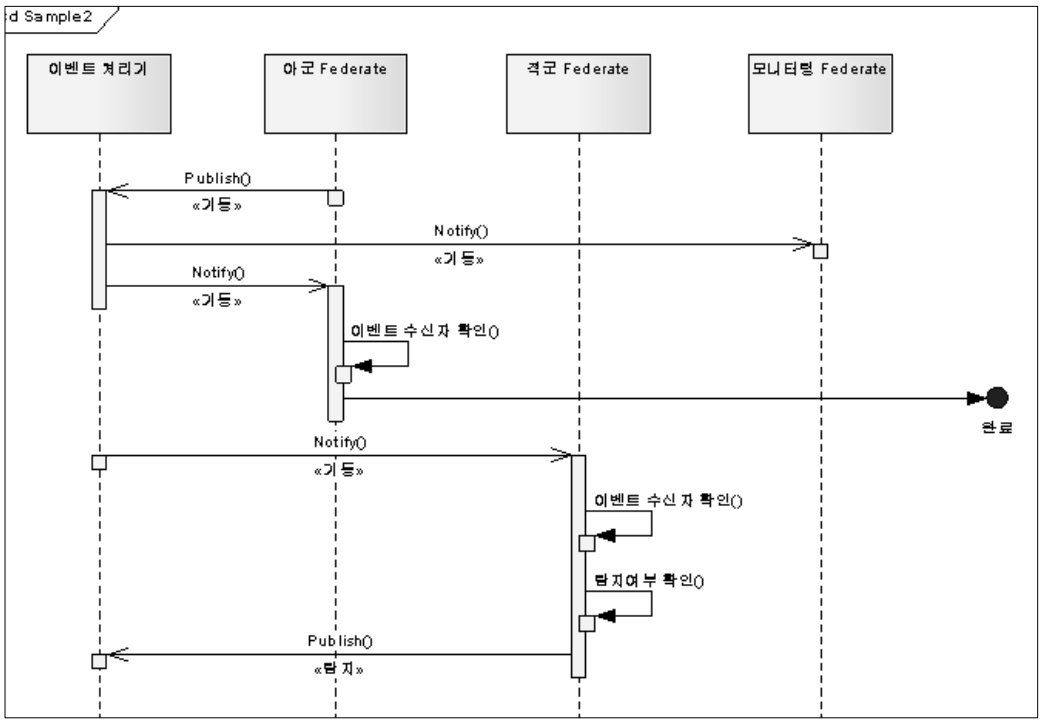


그림 6. 이벤트 처리 Process 예제

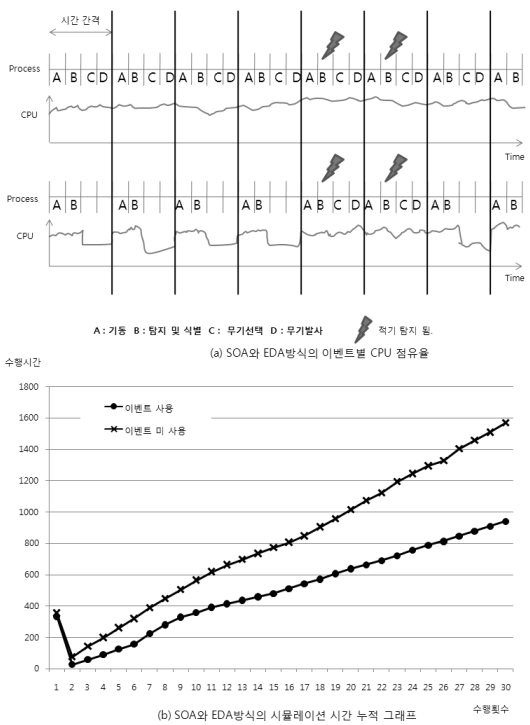


그림 7. SOA와 EDA웹서비스 성능 비교

전", "분석" 이벤트의 수행을 나타내며, 화살표는 적이 탐지되었음을 알리는 표시이다. 그림 7(a)에서 상단 그래프는 SOA방식으로 A, B, C, D 웹서비스들이 순차적으로 조합되어 구축된 수상전 시뮬레이션의 CPU 점유율의 양상이다. 예를 들어, "적탐지" 이벤트의 발생 여부에 상관없이 미리 정의된 기동, 탐지, 교전, 분석과정을 거치게 되므로, 일정한 CPU 점유율을 보이며, 교전이 진행됨에 따라 점유율이 높아지게 된다. 그림 7(a)에서 하단은 EDA방식으로 구현한 수상전 시뮬레이션의 CPU 점유율 변화를 나타낸다. 즉, 적이 탐지되었을 경우에만 교전 및 분석에 필요한 무기선택, 무기발사, 피해분석 과정을 거치게 되므로 CPU 점유율도 "적탐지" 이벤트 발생이후에만 증가한다.

그림 7(b)는 수상전 시나리오의 시뮬레이션 수행 횟수를 증가 시키면서 평균 소요시간을 누적하여 얻은 실험결과이다. 그림 7(b) 그래프에서 보여주듯이 시뮬레이션 모델이 복잡하고, 규모가 커질수록 두 기술을 적용한 수상전 시뮬레이션간의 수행시간 차이가 벌어짐을 알 수 있다.

따라서 이벤트 기술의 적용을 통해 효과적인 모델 구축과 시뮬레이션 수행에 따른 비용을 절감하고 효율적으로 시스템 자원을 운용할 수 있다.

5. 결 론

웹서비스는 워게임 시뮬레이터 구축 시 웹상에서 상용화된 여러 자원을 재사용하고 분산 환경의 여러 페더레이트간의 모의 연동을 지원할 수 있는 매우 효과적인 기술이다. 그러나 기존 SOA방법에서는 웹서비스 기반 페더레이트들이 정해진 순서에 따라 순차적인 경로로 수행되며 새로운 입력에 대한 대응이 진행 중 차단되어, 다양한 전장 이벤트에 따라 여러 경로로 수행될 수 있는 미래 전장 시뮬레이터 구현시 많은 오버헤드를 초래하게 된다. 본 연구에서는 이벤트 기술을 적용한 웹서비스로 페더레이트를 구현하여, 이벤트 수신자에 따라 다양한 흐름으로 시뮬레이션 흐름이 동적.병렬, 비동기적으로 작동되도록 하여 시뮬레이션에 소요되는 자원 및 비용을 절감할 수 있었다.

현재 이벤트 기반 방법론으로 워게임 시뮬레이터를 구축할 경우, 중앙의 이벤트를 감지하고 분배하는 이벤트 처리모듈에 오버헤드가 걸리게 되고, 이벤트를 병렬 처리함에 따라 분산된 이벤트의 발생순서와 도착순서가 반드시 일치한다는 보장이 힘들다는 한계점이 있다. 이를 개선하기 위해 향후, 다양한 워게임 시뮬레이션의 구축에 방법론을 적용해 보고, 방법론을 개선할 계획이다.

참 고 문 헌

1. 최인수, “NCW 환경에서의 국방정보보호 발전방향,” 정보화학회지, 25(9), pp. 81-88
2. 홍윤기, “HLA를 이용한 시뮬레이션의 산업체 도입에 관한 연구,” 한국 국방경영 분석학회 ‘05, 오늘의 국방경영 분석, 21, pp. 59-90, 2005년 3월.
3. Wenguang WANG, “Service-Oriented High Level Architecture,” Simulation Interoperability Standards Organization, 08E-SIW-022, 2008.
4. 권용수, “네트워크 중심 미래전장”, 대한전자공학회 ‘08, 하계종합학술대회, 31(1), pp. 110-114.
5. Möller B, Löf S., “A Management Overview of the HLA Evolved Web Service API,” Simulation Interoperability Standards Organization, 06F-SIW-024, 2006.
6. 이미영, 김명준, “이벤트 기반 서비스 기술 동향,” 전자통신동향분석, 21(5), 2006년 10월.
7. 김기형, 강우석, “A Web Services-based Distributed Simulation Architecture for Hierarchical DEVS Model,” 한국시뮬레이션학회, 제주 international Simulation Multi-conference Part I, pp. 417-426
8. Steve Graham, et al, “Web Services Brokered Notification,” IBM. http://www.ibm.com/developerworks/webservices/library/specification/ws-notification
9. 김병철, 이강선, “시멘틱 웹 서비스를 이용한 워게임 시뮬레이터 제작,” 한국시뮬레이션학회 ‘08, 추계학술대회 논문집, pp. 153-157.
10. Möller B, Löf S., “Mixing Service Oriented and High Level Architectures in Support of GIG[C],” Simulation Interoperability Standards Organization, 05S-SIW-064, 2005.
11. Booth, D. et. al., Web Services Architecture, W3C Working Group Note, 2004, http://www.w3.org/TR/ws-arch/.
12. Web Services Description Language (WSDL) Version 2.0: RDF Mapping, 2007, http://www.w3.org/TR/wsdl20-rdf/
13. 조인호, 주정민, “웹서비스 기반의 분산 기뮬레이션 프로토타입 개발,” 한국경영과학회/대한산업공학회 춘계 공동학술대회, 2005.
14. 이영해, 김숙한, “분산 환경하에서의 웹기반 시뮬레이션에 관한 연구,” 한국시뮬레이션학회 논문지, 7(2), 1998.

이 재 민 (hleejm@mju.ac.kr)

2007 명지대학교 컴퓨터공학과 학사
2007～2009 프람트 재직

관심분야 : 컴퓨터시뮬레이션, 웹서비스, 국방 M&S

김 병 철 (blue7928@mju.ac.kr)

2007 명지대학교 컴퓨터공학과 학사
2009 명지대학교 컴퓨터공학과 석사

관심분야 : 컴퓨터시뮬레이션, 웹서비스, 온톨로지

김 태 섭 (sky_start@mju.ac.kr)

2007 명지대학교 컴퓨터공학과 학사

관심분야 : 컴퓨터시뮬레이션, 웹서비스, 국방 M&S

이 강 선 (ksl@mju.ac.kr)

1992 이화여자대학교 전자계산학과 학사
1994 이화여자대학교 전자계산학과 석사
1998 미) University of Floria, 박사
현재 명지대학교 컴퓨터공학과 교수

관심분야 : 컴퓨터시뮬레이션, 국방 M&S, 웹서비스