무기체계 임베디드 소프트웨어에 대한 TFM 기반 시스템 테스트 모델 설계 및 적용

김 재 환[†] · 윤 희 병[††]

요 약

본 논문에서는 무기체계 임베디드 소프트웨어의 시간 요소를 고려한 TFM(Time Factor Method) 기반의 시스템 테스트 모델을 설계하고, 적용사례를 통하여 결과를 제시한다. 이를 위해 무기체계 임베디드 소프트웨어의 특징과 시스템 테스트 그리고 객체지향 모델의 표현방법인 UML 표기법에 대하여 알아보고, 시스템 테스트 모델 설계를 위한 TFM 접근 방법으로 시간 요소를 고려한 테스트 방법과 시간 요소 측정 방법 그리고 테스트 케이스 선정 알고리즘을 제시한다. 무기체계 임베디드 소프트웨어의 TFM 기반 시스템 테스트 모델은 세 가지 요소 (X,Y,Z)로 구성되며, "X" 에서는 최대시간경로를 선정하는 알고리즘을 통해 테스트 케이스가 도출되고, "Y" 에서는 Sequence Diagram과 관련된 객체를 식별하고, "Z"에서는 Timing Diagram을 통하여 식별된 각 객체들의 실행시간을 측정한다. 또한 제안한 TFM 기반 시스템 테스트 모델을 "다기능 미사일 방어시스템"에 적용하여 테스트 케이스를 추출하는 방법을 제시한다.

키워드 : TFM, 객체, 무기체계 소프트웨어, 시간요소, 시스템 테스트 모델

Design and Application of the TFM Based System Test Model for the Weapon System Embedded Software

Jaehwan Kim[†] · Heebyung Yoon[††]

ABSTRACT

In this paper we design the system test model for the weapon system embedded software based on the Time Factor Method(TFM) considering time factors and suggest the results through the case study. For doing this, we discuss the features, system tests and the object-oriented model based UML notations of the weapon system embedded software. And we give a test method considering time factors, a measuring method to time factors, and a test case selection algorithm as an approach to the TFM for designing the system test model. The TFM based system test model consists of three factors (X, Y, Z) in the weapon system embedded software. With this model, we can extract test cases through the selection algorithm for a maximum time path in "X", identify the objects related to the Sequence Diagram in "Y" and measure the execution time of each objects which is identified by the Timing Diagram in "Z". Also, we present the method of extracting the system test cases by applying the proposed system test model to the "Multi-function missile defense system".

Key Words : Time Factor Method, Object, Weapon System Embedded Software, Time Factor, System Test Model

1. 서 론

무기체계 임베디드 소프트웨어는 HW 중심에서 SW 중심으로 개발기술이 발전하고 있으며 최신예 기종인 F-22는 80%를 소프트웨어로 처리하는 등 소프트웨어의 복잡도가 증가하고, 이에 따라 개발비용 및 개발시간이 급격한 증가를 보이고 있다[1]. 이러한 문제를 해결하기 위해 재사용성을 고려한 객체지향 개발 방법론으로 소프트웨어를 개발하는 추세에 있으며, 객체지향의 모델링 언어인 UML이 설계 도구로 표준화가 진행 중에 있다[2].

한편, 무기체계 임베디드 소프트웨어의 테스트를 위해서는 일반 소프트웨어와는 달리 기능이나 로직(Logic)은 물론 시간제약 조건을 반드시 준수하여야 한다. 이렇듯 무기체계 임베디드 소프트웨어를 테스트하기 위해서는 무기체계의 특징을 고려한 테스트 기법이 요구되고 있다.

테스트 측면에서 보면 시스템 테스트는 요구사항 설계서를 기초로 하여 설계서에 명시된 기능이 제대로 실현되는가를 검증한다. 또한 요구사항 명세서의 내용을 기준으로 테스트 데이터를 생성하며, 성능관점에서 검사를 실시한다[3]. 최근까지의 소프트웨어 신뢰성을 보장하기 위한 시스템 테스트 기법으로는 통계학적인 접근방법[4], 인공지능적인 접

† 준 회 원 : 국방대학교 전산정보학과 석사과정
†† 종신회원 : 국방대학교 전산정보학과 부교수
논문접수 : 2006년 8월 8일, 심사완료 : 2006년 9월 29일

근방법[5], 실시간 분석 테스트방법[6], 결함특성 분석기법[7] 그리고 기능을 그룹별로 구분하여 테스트[8]를 하는 기법등 많은 연구가 진행되었다.

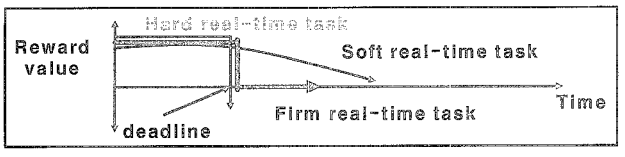
이에 본 논문에서는 시간 요소를 고려한 테스트 방법과 시간 요소 측정 방법 그리고 테스트 케이스 선정 알고리즘을 활용한 TFM(Time Factor Method) 접근 방법을 제시하고, 이를 토대로 UML 표기법과 무기체계의 시간제약 특징을 고려한 무기체계 임베디드 소프트웨어의 시스템 테스트 모델을 (X,Y,Z)의 세 가지 요소를 활용하여 제안한다. 그리고 제안한 TFM 기반 시스템 테스트 모델을 "다기능 미사일 방어시스템"의 사례연구를 통하여 테스트 케이스 추출방법을 제시한다.

2. 관련 연구

2.1 무기체계 임베디드 소프트웨어의 특징

무기체계는 Critical System으로 분류되고, 대부분 임베디드 소프트웨어로 구성되어 있으며 시간제약 조건이 준수되어야 한다[9]. 임베디드 소프트웨어의 특징 중 하나인 시간제약 조건은 무기체계에서 중요한 의미를 가지며, 종료시간(Deadline)의 준수는 매우 중요한 특징 중의 하나이다. (그림 1)은 무기체계 임베디드 소프트웨어의 시간제약 조건을 나타내고 있다.

(그림 1)의 시간제약 조건 중에서 무기체계 임베디드 소프트웨어에 가장 적합한 것이 경성 종료시간(Hard Real-Time Task)으로 실시간 긴급을 요하며, 실패 시 막대한 인명이나 재산의 피해를 초래할 수 있기 때문에 소프트웨어 개발 시 중요한 요건으로 간주된다. 그러므로 무기체계 임베디드 소프트웨어 테스트 분야에서는 시간요소가 중요한 점검 항목으로 평가된다.



(그림 1) 무기체계 임베디드 소프트웨어의 시간제약 조건

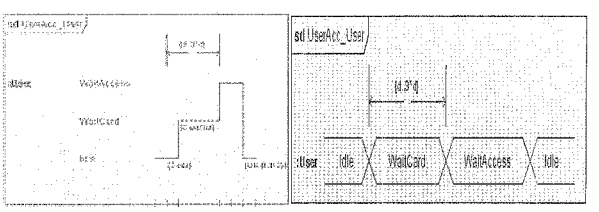
2.2 소프트웨어 시스템 테스트

소프트웨어 테스트에는 단위 테스트, 통합 테스트, 시스템 테스트 등이 있는데, 본 연구에서는 단위/통합 테스트는 완료된 것으로 가정하였다. 시스템 테스트에는 ASF(Atomic System Function) 개념이 있는데, ASF는 입력 이벤트에서 시작해서 출력 이벤트까지의 특정 단위를 의미하며, 클래스 테스트 케이스 추출기법, 통합 테스트 케이스 추출기법과 시스템 테스트 케이스 추출기법에서 널리 이용되고 있다[3][8]. 시스템 테스트에서 ASF는 발생할 수 있는 하나의 기본 단위이며, 하나의 동작이 시작하여 마무리 되는 하나의 실행 결과라 할 수 있다. 시스템 테스트와 시간과의 관계에 대한 국내·외 연구진행을 알아보면, 대부분 실시간 시스템의 연속된 이벤트 발생에 관한 연구와 실시간에 따른 사건중심에 대한 연구를 진행 중이며, 객체를 하나의 객관적인 테스트 요소로서 테스트에 활용하는 사례는 다소 미진한 상태이다.

2.3 UML 상호작용 다이어그램

UML은 1980년대 후반에서 1990년대에 이르는 기간 동안 나타난 객체지향 분석 설계 방법론의 흐름을 이어받은 객체지향 모델링 언어이다. 1999년 3월 UML은 OMG(Object Management Group)에 의해서 표준으로 확정되었으며 지속적인 개정 작업이 이루어지고 있다[10].

UML은 크게 Dynamic Behavior Diagram과 Static Structure Diagram으로 구분되는데, 이 같은 Diagram 중에 시스템 테스트와 연관이 있는 다이어그램은 상호작용 다이어그램 (Interaction Diagram)으로 그 종류에는 Sequence Diagram, Collaboration Diagram, Timing Diagram 등이 있다. Sequence Diagram은 Collaboration Diagram과 함께 시스템의 동적구조, 즉 객체와 객체사이, 객체와 객체그룹사이, 객체그룹과 객체그룹 사이의 동적인 행위를 기술한다[11,12]. 또한 UML 2.0에서는 새로 추가된 형태의 Timing Diagram을 (그림 2)와 같이 두 가지 형태로 제공한다. 본 논문에서는 경제적이고 간단한 표현인 우측 다이어그램을 사용한다.



(그림 2) UML Timing Diagram

3. 시스템 테스트 모델 설계

3.1 시간 요소를 고려한 테스트 방법 (TFM)

하나의 소프트웨어는 각기 다른 객체와 객체의 결합이라고 할 수 있으며, 시스템 테스트는 이들 객체들 간의 결합이 제대로 연결되는가를 검증하는 절차이다. UML에서는 이러한 연결이 객체 간 메시지의 흐름을 통해서 이루어진다[13].

또한 각각의 객체는 인터페이스를 통해 고유한 성능을 나타내며 "0"이 아닌 고유의 실행시간(Execution Time)을 갖고 있다. 아래의 수식 (1)에서 보는 것과 같이 실행시간(CPUet)은 프로그램에 대한 CPU 클럭 주기(CPUCl)와 클럭 주기 시간(Ct)의 곱이 되며, 클럭 주기 시간은 컴퓨터에 장착된 CPU에 따라 제공되기 때문에 고유의 클럭 주기율(Cr)로 나누어서 구할 수 있다. 또한 수식 (2)에서 보는 것과 같이 CPU 클럭 주기는 각각의 클래스에 대한 명령어 수(Ci)와 CPI(Clock Per Instruction)를 곱하여 전체의 합으로 구할 수 있다[14].

$$CPUet = CPUCl \times Ct = \frac{CPUCl}{Cr} \quad (1)$$

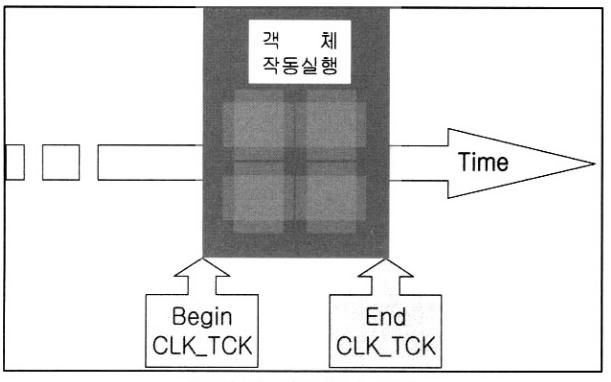
$$CPUCl = \sum_{i=1}^{n} (CPIi \times Ci) \quad (2)$$

즉, 하나의 ASF는 시스템 측면에서 보면 입력 이벤트에서 시작하여 출력 이벤트까지의 특정 단위로, 하나의 기능을 수행하기 위한 고유한 특성을 갖는 여러 객체들의 집합이라 할 수 있으며, ASF의 실행시간은 모든 관계된 객체의 실행시간을 합한 총 시간이 된다.

객체의 실행시간은 처음부터 일정한 시간으로 정해지는 것도 있지만 입력을 기다리는 경우도 있고, 또한 데이터 처리에 따라 시간이 유동적일 경우도 있다. 그러한 경우에는 스트레스 테스트 방법에 따라 최대시간을 기준으로 측정된다[15]. 그것은 최악의 경우에도 제한시간을 준수해야 하는 무기체계 임베디드 소프트웨어의 특징을 만족시키기 위함인데, 같은 경로를 진행한다고 해도 처리시간이 유동적일 경우는 최대한 오래 진행될 경우의 수를 측정해야 모든 상황을 포함할 수 있기 때문이다. 이러한 시간측정은 실제 작동될 프로세서에서 실행되어야 하는데 그것은 작동할 프로세서의 처리속도에 따라 처리시간(능력)이 달라질 수 있기 때문이다.

3.2 시간 요소 측정 방법

객체의 시간 요소는 TFM 기반 모델에서 중요한 요소로 작용하는데, UML의 Timing Diagram에서 제공하면 그대로 사용하면 되겠지만, 제공하지 않으면 측정에 의한 값을 사용하여야 한다. (그림 3)은 C++에서 제공하는 기능으로 프로세스 혹은 기능(Function)이 실행된 후 시간이 얼마나 경과하였는지를 알아내는 방법이다. 객체 혹은 객체의 집합으로 이루어진 기능을 실행하기 전에 CPU 칩 내에서 제공하는 CLK_TCK를 측정하고 객체가 완료된 후에 다시 CLK_TCK를 측정하여 진행된 시간차를 확인하면 작동시간을 파악할 수 있다. 이러한 원리를 이용하여 객체 시작 전과 작동 완료 후에 측정된 값을 시간으로 환산하면 된다.



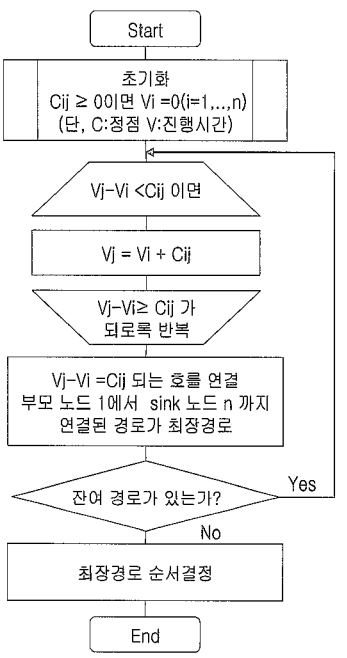
(그림 3) 시간 요소 측정

이러한 시간 요소 측정은 개발하려는 시스템의 CPU와 동일한 환경에서 측정하여야 하며, 추가로 고려되어야 할 사항은 객체 간의 자료 송·수신에 필요한 대기 시간이 고려되어야 한다. 이러한 모든 시간 요소는 시스템 테스트에서 제한시간의 요소로 작용한다.

3.3 테스트 케이스 선정 알고리즘

무기체계는 성능뿐만 아니라 제한된 시간 내에 성능을 발휘해야 그 의미가 있다. 아무리 좋은 성능을 가지고 있다 하여도 시간제약 사항은 반드시 지켜져야 하는 것이다. 최대시간 측정을 위한 알고리즘으로 Dijkstra의 최단경로문제 알고리즘[16]의 기본 프로세스를 응용한 최대시간 측정 알고리즘을 (그림 4)에 제시하였다.

최대시간 측정 알고리즘의 원리는 정점과 정점 간의 진행시간을 측정할 경우 이전 값을 계속 저장하면서 진행하고 가능한 정점을 모두 진행하여 이전 값과 비교하여 제일 큰 값을 찾아가는 방식이다. 위의 알고리즘을 통하여 최장시간 경로를 도출해 낼 수 있으며 모든 경우의 수를 위해서 잔여 경로에 대한 경로 검색을 하면 그 경로 수만큼의 시간경로 즉, ASF가 식별된다.



(그림 4) 최대시간 측정 알고리즘

3.4 TFM 기반 시스템 테스트 모델 설계

TFM 기반 모델 설계를 위해 시간 관점을 기준으로 (그림 5)와 같은 방법으로 테스트 케이스를 도출한다. 시스템 테스트의 특징을 고려한 ASF 활용을 위해서 Collaboration Diagram에 최대시간 측정 알고리즘을 적용하여 ASF를 도출하였고, 무기체계의 특징을 고려한 TFM 활용을 위해서 실시간 응답성을 적용한 Sequence Diagram과 Timing Diagram을 활용한다.

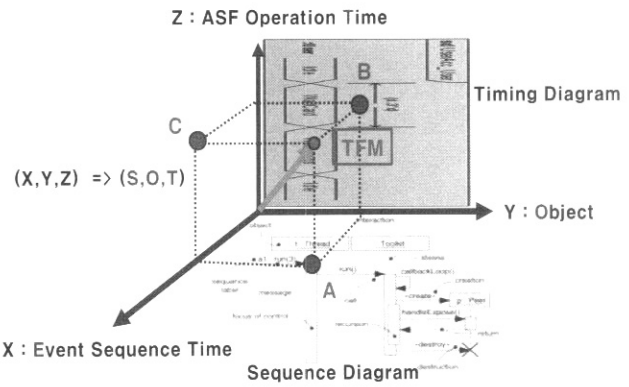
테스트 케이스 선정 알고리즘을 UML에 적용하고 Diagram 간의 관계를 파악하기 위해 TFM(Time Factor Method) 기반 시스템 테스트 모델을 설계하면 다음과 같다. 우선 TFM은 (그림 6)과 같이 세 가지의 요소를 가지고 있다. 즉, (X,Y,Z)로 구성되는데, "X"에서는 최대시간경로를 선정하여 테스트 케이스가 도출되고, "Y"에서는 Sequence Diagram에서 관련된 객체를 식별하고, "Z"에서는 식별된 각 객체의 실행시간을 측정한다.

(그림 6)을 보면 Sequence Diagram과 Timing Diagram의 객체(Object)를 공통분모로 하여 배치하며, X축과 Y축의 평면에는 Sequence Diagram을 배치하여 X축에는 Event Sequence Time을 통한 메시지 흐름을 파악하고, Y축에는 Sequence Diagram에 관계된 Object를 식별한다. Y축과 Z축에는 Timing Diagram을 배치하여 Y축에는 Sequence Diagram의 공통분모를 Timing Diagram의 Object와 일치시키고, Z축에는 ASF의 Operation Time(Object 실행시간)을 배치한다.

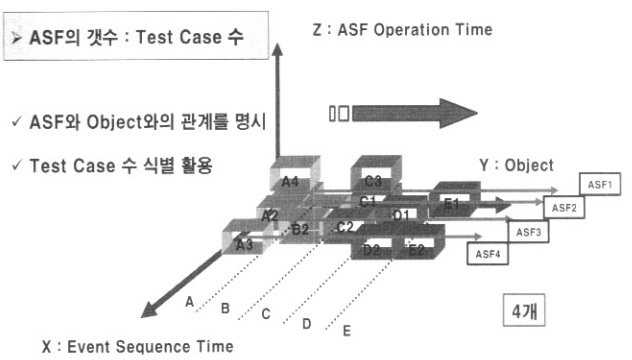
이 모델의 공통분모는 Y축의 Object로서 UML에서 가장 기본적이고 중요한 요소를 활용하는 것이다. 이 모델을 이해하기 위해 (그림 7)과 같이 하나의 예시 소프트웨어를 통하여 보면 각 객체 간의 관계를 파악할 수 있고, 재사용과 중복 사용되는 객체들 사이의 관계를 이해하는데 용이하다.

(그림 7)에서 예시 소프트웨어의 객체 구성을 살펴보면 Object가 "A" 부터 "E" 까지 종류별로 사용되어있다. "A" Object는 4번(A1~A4) 사용되어있고, "B" Object는 2번, "C" Object는 3번, "D" Object는 2번, "E" Object는 2번으로 각각 사용되었음을 알 수 있다. Object가 Z축으로 층을 쌓으면서 올라가 있는 경우(A4,C3)가 있는데 이 경우는 같은 Object가 하나의 ASF에 재사용되어 구성됨을 알 수 있다.

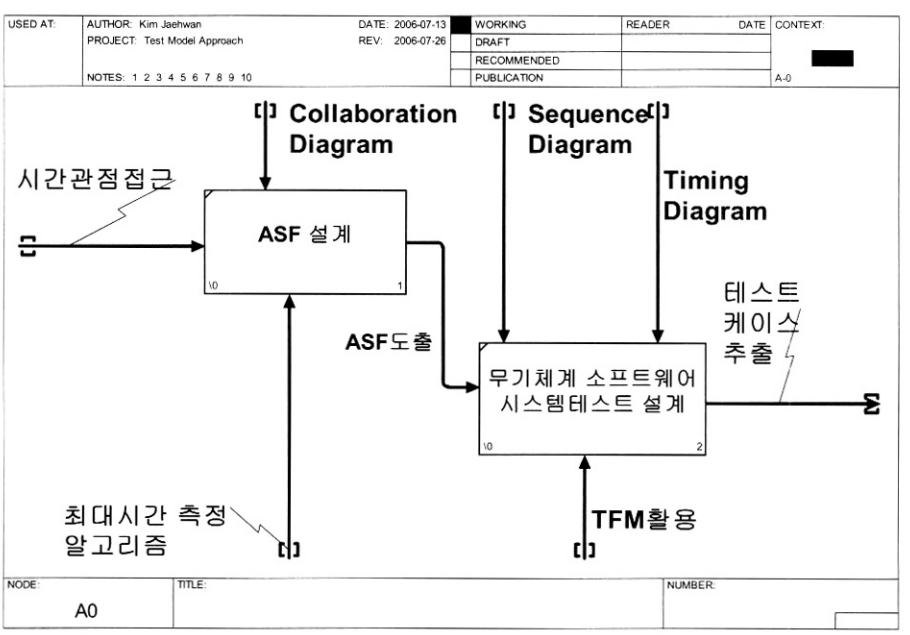
따라서 예시 소프트웨어에서 사용된 총 Object는 13번임을 알 수 있으며, X축과 Y축 선상의 Object의 합이 하나의 ASF가 된다. 예를 들어, ASF1은 객체가 A1,A4,C1,C3,E1등



(그림 6) TFM 기반 시스템 테스트 모델 설계



(그림 7) 예시 소프트웨어의 테스트 케이스 식별

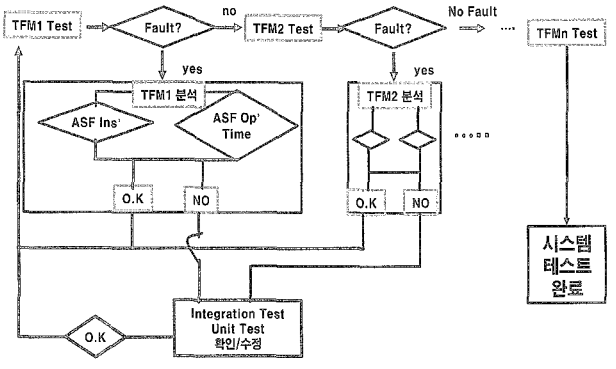


(그림 5) 시스템 테스트 모델 접근 방법

으로 구성되어 있으며, 예시된 소프트웨어는 총 4개의 ASF로 구성되어 있음을 알 수 있다. 또한 X,Y,Z 축을 통하여 각각의 ASF와 Object와의 관계 알 수 있고, ASF를 통한 테스트 케이스를 파악할 수 있다.

3.5 TFM 기반 시스템 테스트 모델 적용방법

제안한 TFM 기반 시스템 테스트 모델의 적용방법은 (그림 8)과 같다. 먼저 TFM 개수가 테스트 케이스가 되고, 테스트 케이스 첫 번째인 TFM1을 테스트한다. 정상작동 상태(ASF1)와 작동시간(ASFt1)을 점검한다. 이때 이상이 없으면 TFM2 테스트를 실시한다. 만약에 TFM1에서 결함이 발생하면 두 가지의 경우를 생각해 볼 수 있는데, 첫째는 설계단계에서 객체를 구성할 때에 제한시간을 미 고려한 설계에 의한 ASF의 시간초과 문제가 될 수 있으며, 둘째로 설계는 기준 제한시간 내에 잘 구성되었으나 구성된 객체간의 인터페이스 상의 문제가 발생한 경우이다. 결함수정은 TFM 기반 모델을 통해서 ASF 작동상의 문제점을 혹은 ASFt1 문제를 파악하여 수정한다. 수정 후 다시 TFM1 테스트를 실시하고 다음 단계인 TFM2로 진행한다. TFM2에서도 TFM1과 같은 동일한 절차를 진행하여 식별된 모든 TFMn 테스트가 끝나면 시스템 테스트는 완료된다.



(그림 8) TFM 기반 시스템 테스트 모델 적용방법

4. 사례 연구

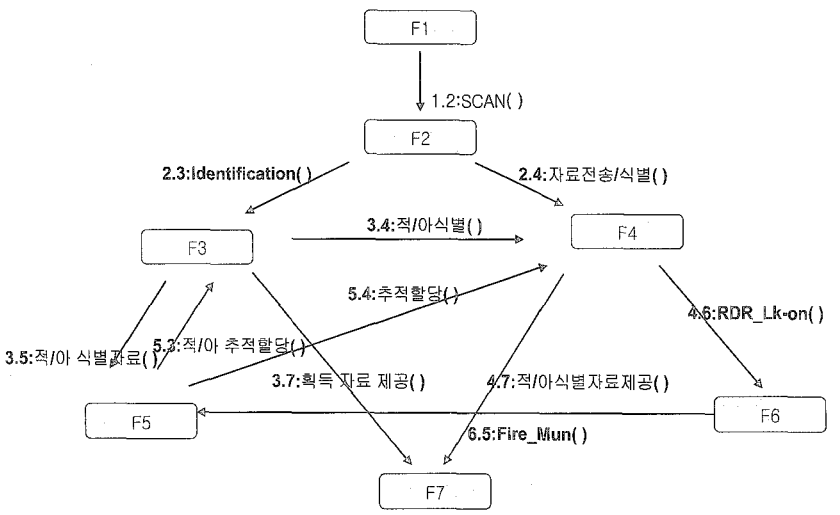
TFM 기반의 시스템 테스트 모델에 대한 사례연구로 "다기능 미사일 방어시스템"[8]을 이용하였다. 이 시스템은 제한된 미사일 방어시스템으로 (그림 9)와 같이 서로 매우 복잡한 관계를 갖고 있다. 단위기능 객체는 총 7개로 구성되어 있으며 각 객체는 서로 여러 기능의 관계를 형성하고 있어서 경로를 통한 테스트 케이스는 무수히 많게 된다. 이러한 경우 모든 경우의 수를 테스트하게 되면 비효율적인 테스트가 되고, 일부 테스트 케이스 누락은 신뢰성을 보장할 수 없다. 또한 예시 사례는 제한적인 모델로서 실제 무기체계에서는 보다 복잡하고 많은 테스트 케이스가 도출될 것이다. 그러한 모든 경우의 수를 테스트 하게 되면 비용과 시간은 급격히 증가한다. 이에 본 논문에서 제안한 방법으로 테스트 케이스를 도출하면 다음과 같다.

객체 실행시간이 "0"으로 존재하는 경우는 없으므로 <표 1>과 같이 가정하여 부여하였다.

우선 ASF를 식별하기 위해서는 Collaboration Diagram에 최대시간 측정 알고리즘을 통하여 ASF를 식별할 수 있다. (그림 10)은 식별된 객체의 최대시간 경로를 나타내고 있다. 이 시스템을 3.4절에서 제안한 TFM 기반 시스템 테스트 모델에 적용하여 테스트 케이스를 추출할 수 있다.

〈표 1〉 미사일 방어시스템의 객체 실행시간

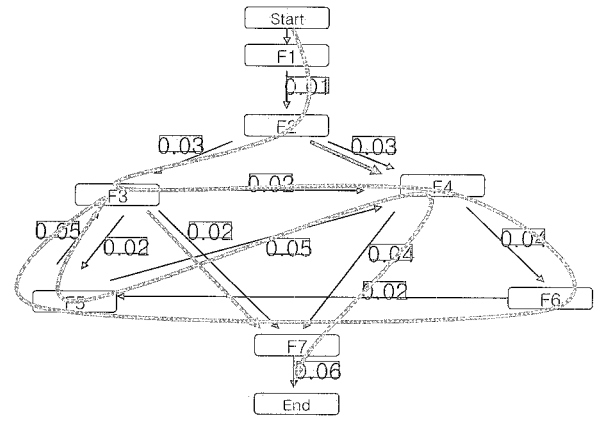
객체	설 명	시간(s)
F1	Basic Scanning Examined Area	0.01
F2	Basic Scanning and Maintenance of TGT	0.03
F3	Processing on Data on TGT	0.02
F4	Dynamical Maintenance of TGT (Track-Assign)	0.04
F5	Multi-TGT, Multi-Track Assignment	0.05
F6	Extended Fire Control (Assignment of Rockets)	0.02
F7	Deletion of Non Dangerous TGT	0.06



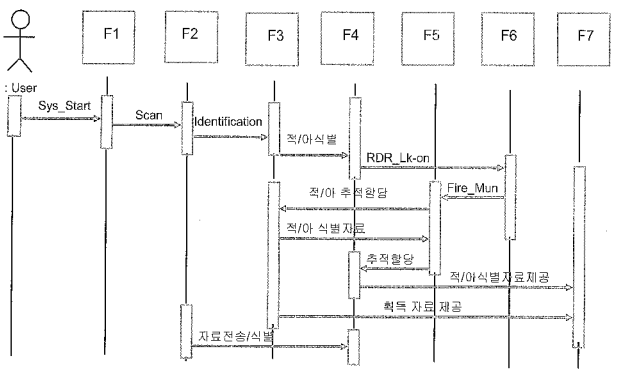
(그림 9) 다기능 미사일 방어시스템의 Collaboration Diagram

(그림 11)은 식별된 ASF를 활용한 Sequence Diagram을 나타내고 있다. 식별된 ASF는 세 가지 경우로 각각의 시간 흐름을 알 수 있다.

적용사례를 기준으로 Timing Diagram을 위해서는 간단하고 경제적인 표현을 활용하여 F1부터 F7까지 작동시간을 기준으로 만든다. (그림 12)는 Sequence Diagram과 Timing Diagram을 적용한 TFM 기반 시스템 테스트 모델의 적용사례이다.



(그림 10) 식별된 객체 최대시간 경로



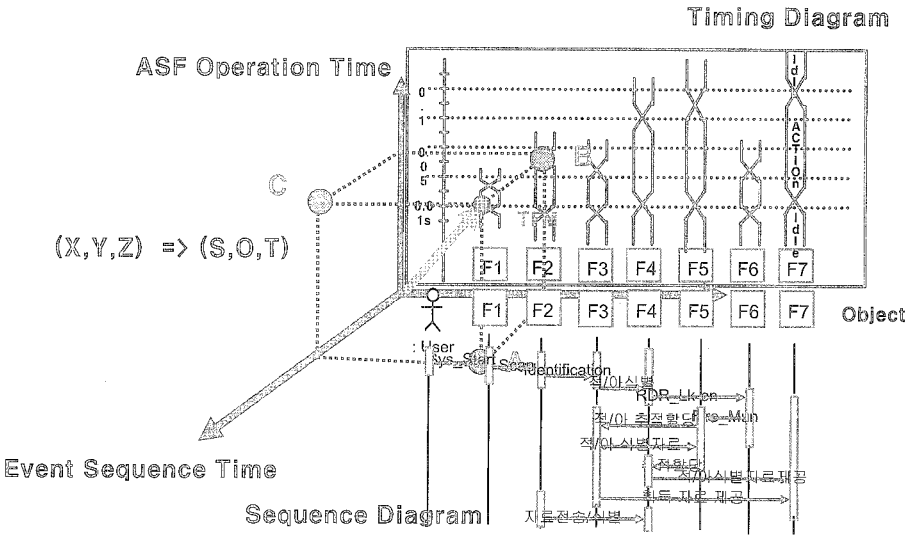
(그림 11) 식별된 Sequence Diagram

TFM 기반 모델을 통하여 반복 재사용한 객체에 대한 식별이 가능하고 객체 서로 간의 상관관계를 알 수 있다. 또한 TFM 기반 모델의 X와 Y축의 Sequence Diagram과 최대시간 측정 알고리즘을 통한 결과는 <표 2>와 같이 세 가지의 테스트 케이스가 식별되었다.

사례 연구에서 제시된 무기체계 임베디드 소프트웨어의 테스트는 <표 2>의 세 가지 테스트 케이스를 확인하면 된다. 예를 들어, 첫 번째 경로의 테스트 케이스는 TFM1으로 경로는 F1->F2->F3->F4->F6->F5->F3->F5->F4->F7 순으로 테스트를 하고 제한시간은 0.34(s)가 된다. 이와 같은 방법으로 TFM2와 TFM3까지 테스트하면 시스템 테스트는 완료된다. 만약 제한시간을 초과 한다면 설계단계에 잘못이 있는지 혹은 인터페이스 상에 문제가 있는지 확인이 필요하고, 이에 따른 시스템 테스트 하위의 테스트를 다시 고려해 볼 필요도 있다.

얻은 결과에 대하여 기존 테스트 방법과 비교를 위해 기능을 중심으로 그룹화하여 테스트 케이스를 도출한 "다기능 미사일 방어시스템"[8]의 사례연구와 비교하였다. <표 3>에서 보는바와 같이 기능중심의 테스트 케이스는 총 4개의 테스트 케이스가 식별 되었으며, 본 연구에서 제안한 방법인 TFM 기반 시스템 테스트 모델을 활용하여 테스트한 결과 테스트 케이스는 3개가 도출되었다.

제시된 방법의 장점은 무기체계 특성 중의 하나인 시간제약 사항을 만족하면서 동시에 기능을 점검할 수 있으며, 객체 실행시간을 통한 테스트 케이스의 수도 줄일 수 있다. 또한 객체의 정상작동 여부와 객체 간의 인터페이스 확인도 가능하다. 개별적으로 잘 작동하던 객체가 통합 상태에서 문제가 발생하면 로직 혹은 인터페이스 문제일 수 있으며 시간은 무한대 혹은 초과시간으로 측정됨으로써 결함을 확인할 수 있다. TFM 기반 시스템 테스트 모델에서 측정한



(그림 12) TFM 기반 시스템 테스트 모델 적용사례

〈표 2〉 식별된 테스트 케이스

테스트 케이스	경 로	제한시간(s)
1	F1->F2->F3->F4->F6->F5->F3->F5->F4->F7	0.34
2	F3->F7	0.08
3	F2->F4	0.07

〈표 3〉 테스트 결과 비교

구 분	기능중심 테스트	TFM 테스트
테스트 케이스	1: F1->F2->F3->F4->F6->F5->F3 2: F2->F4->F7 3: F3->F7 4: F3->F5->F4	1: F1->F2->F3->F4->F6->F5->F3->F5->F4->F7 2: F3->F7 3: F2->F4
특 징	기능을 그룹화하여 테스트 실시	객체를 통한 시간중심 테스트로 시간과 기능을 동시에 테스트

시간보다 많은 시간이 소요되는 경우에는 잠재적인 결함 가능성도 내재된 것으로 판단할 수 있다. 또한 작동시간이 너무 빠르거나 늦으면 객체의 누락이나 인터페이스상의 문제가 있을 수 있다.

5. 결 론

본 논문에서는 시간요소를 고려한 무기체계 임베디드 소프트웨어의 시스템 테스트 케이스 선정을 위하여 시스템 테스트의 특징과 무기체계의 시간제약을 고려한 응답성 그리고 UML 표기의 Diagram을 분석하여 시스템 테스트 모델인 TFM 기반 시스템 테스트 모델을 설계하였다. 또한 TFM 기반 시스템 테스트 모델을 활용하여 "미사일 방어시스템"의 시스템 테스트 케이스를 도출하였다. 또한 결과에 대하여 기존의 기능중심의 테스트 방법과 본 연구에서 제안한 TFM 기반 시스템 테스트 모델의 결과를 비교한 결과 테스트 케이스가 줄었음을 확인 하였다.

제안한 TFM 기반 시스템 테스트 모델을 통하여 무기체계 특성 중의 하나인 시간제약 사항을 만족하면서 동시에 로직을 점검할 수 있었으며, 객체 실행시간을 통한 테스트 케이스의 수도 줄일 수 있었다. 또한 객체 간의 인터페이스도 확인이 가능하였다.

향후 연구로는 본 논문에서 제안한 TFM 기반 시스템 테스트 모델의 실제 적용을 위한 자료 송·수신에 필요한 Bus 시간 적용에 대한 연구가 필요하다.

참 고 문 헌

[1] Software Technology Support Center, *Guidelines for Successful Acquisition and Management of Software-Intensive Systems(GSAM)*, Department of the Air Force, Ver. 3.0, 2000. 5.

[2] Object Management Group, "UML 2.0 Superstructure RFP," OMG document no. pct/03-08-02, 2003. 2.

[3] Paul C. Jorgensen, *Software Testing, A Craftsman's Approach*, Part IV, CRC Press, 1995.

[4] Yan Jiong, Wang Ji and Chen Huowang, "Deriving software statistical testing model from UML model," 3rd International Conference Quality Software, pp.343-350, 2003. 11.

[5] S.D. Miller, R.A. DeCarlo, and A.P. Mathur, "A software cybernetic approach to control of the software system test phase," COMPSAC, Vol.2, pp.103-108, 2005. 7.

[6] R. Fryer, "A FPGA based real-time analyzer for in-flight software & system testing," DASC, Vol.2, pp.506-506, 2004. 10.

[7] C. Yilmaz, M.B. Cohen, and A.A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces," IEEE Transactions on Software Engineering, Vol.32, Is. 1, pp.20-34, 2006. 1.

[8] Mark Sh. Levin and Mark Last, "Muli-Function System Testing: Composition of Test Sets," 8th IEEE International Symposium on High Assurance Systems Engineering(HASE'04), pp.99-108, 2004.

[9] A. En-Nouaary, F. Khendddek, and R. Dssouli, "Testing Embedded Real-Time Systems," 7th International Conference on Real-Time Computing Systems and Applications(RTCSA'00), pp.417-424, 2000.

[10] B. Selic, "Tutorial h2: an overview of UML 2.0," 25th International Conference Software Engineering, pp.755-756, 2003.5.

[11] Sang-Uk Jeon, Jang-Eui Hong and Doo-Hwan Bae, "Interaction-based behavior modeling of embedded software using UML 2.0," IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing(ISORC'06), p.5, 2006.4.

[12] M. Bjerkander and C. Kobryn, "Architecting Systems with UML 2.0," IEEE Software, Vol. 20, Is. 4, pp.57-61, 2003. 9.

[13] Rumbaugh, Jacobson, and Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

[14] David A. Patterson and John L. Hennessy, *Computer Organization And Design*, ELSEVIER, 2005.

[15] M. Gh, Mohammad and K.K. Saluja, "Optimizing program disturb fault tests using defect-based testing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.24, Is. 6, pp.905-915, 2005. 6.

[16] William Stallings, *Data and Computer Communications* 7th Ed., Pear Education, 2004.

김 재 환

e-mail : kimch092@yahoo.co.kr
1993년 공군사관학교 전자계산학과(학사)
2005년~현재 국방대학교 전산정보학과 (석사과정)
관심분야 : 임베디드 소프트웨어, 테스팅

윤 희 병

e-mail : hbyoon@kndu.ac.kr
1983년 해군사관학교(이학사)
1986년 연세대학교(공학사)
1991년 미국 해군대학원 전산공학(석사)
1998년 미국 Georgia Institute of Technology 전산공학(박사)
2002년~현재 국방대학교 전산정보학과 부교수
관심분야 : 임베디드 소프트웨어, 소프트웨어 공학, 소프트웨어 테스팅