# TMO 기반의 정적 분석 도구를 위한 PS-Block 구조 설계

김윤관, 신원, 김태완, 장천현
건국대학교 컴퓨터 공학과
{apostlez, wonjjang, twkim, chchang}@konkuk.ac.kr

# Design of PS-Block Structure for TMO Model based Static Analysis Tool

Yun-Kwan Kim, Won Shin, Tae-Wan Kim, Chun-Hyon Chang
Dept of Computer Engineering, Konkuk University

요 약

실시간 시스템은 시간적 정확성을 갖기 때문에 소형 임베디드 시스템부터 대형 분산 시스템까지 많은 분야에서 사용되고 실시간 시스템을 기반으로 하는 실시간 프로그램도 많은 분야에서 사용되고 있다. 이러한 실시간 프로그램의 시간적 특성을 지키기 위해 개발자들은 프로그램 개발에 집중하지 못하고 실행시간의 정의와 정의한 실행시간의 정확성 검사에 많은 시간을 보내고 있다. 실시간 시스템에 대한 연구 결과로서 TMO 모델은 실시간 개념에 따른 시간 처리의 다양한 기능을 지원하고, 응답시간을 보장하여 개발자가 프로그램 개발에 집중할 수 있다. 하지만, 실행시간의 정의는 개발자에 의해 이루어지기 때문에 이를 정의하고 그 정확성 여부를 확인하는 작업은 어렵다. 이러한 문제로 인하여 실행시간 정의의 기준점을 제시할 수 있는 도구가 필요하지만 이를 위한 TMO 분석 도구에 대한 연구는 미흡하다. 이에 본 논문에서는 TMO 기반 정적 분석 도구를 위한 PS-Block을 제시한다. PS-Block은 블록 단위로 실행시간을 분석할 수 있는 기반으로써 프로그램을 작업 단위로 분리하여 분석할 수 있도록 한다. 이를 기반으로 실행시간을 분석하여 시간 정보 결정의 기준으로 하고, 실시간 메소드의 적시성 확인을 쉽게 함으로써 실시간/신뢰성의 향상과 개발 기간을 단축할 수 있다.

## 1. 서론

실시간 시스템은 시간적 정확성을 가지고 동작하기 때문에 소형 임베디드 시스템부터 대형 분산 시스템까지 많은 분야에서 사용되고 있고 앞으로도 사용 분야가 더욱 넓어질 수 있는 컴퓨터 분야이다. 이러한 실시간 시스템의 확대에 따라 실시간 시스템을 기반으로 하는 실시간 프로그램에 대한 사용도 늘어나고 있다. 실시간 프로그램은 주어진 시간에 정확하게 일을 수행해야 하는 특성을 가지고 주어진 시간을 지키지 못하는 경우 경제적 피해를 주거나 인명사고로 이어질 수 있다. 제한 시간을 벗어나는 문제는 개발자의 실시간 개념 부족이나 시스템의 시간 지연 요소에 의해 발생할 수 있다. 따라서 실시간 프로그램 개발자들은 프로그램을 개발할 때에 프로그램 실행시간의 정의와 정의한 실행시간의 정확성 여부 검사로 개발에 전념하지 못하고 프로그램 분석에 많은 시간을 보낸다.

이러한 문제들 때문에 실시간 시스템에 대한 많은 연구가 이루어 졌고 그 결과로 TMO(Time-triggered Message-triggered Object) 모델은 실시간 개념에 따른 시간 처리에 다양한 기능을 지원하고 응답시간을 보장하여 개발자가 프로그램 개발에 집중할 수 있다. 하지만 실행시간의 정의는 개발자에 의해 이루어지기 때문에 이를 정의하고 정의된 실행시간의 정확성 여부를 확인하는 작업에는 여전히 어려움이 있다. 이러한 문제로 인하여 실행시간 정의의 기준점을 제시할 수 있는 도구가 필요하지만 이를 위한 TMO 분석 도구에 대한 연구는 미흡하다

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음

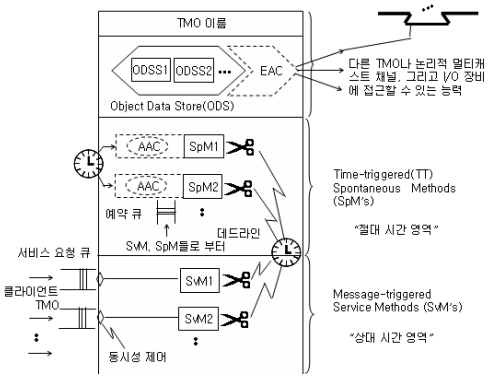
[1]. 이에 본 논문에서는 TMO 기반의 정적 분석 도구를 위한 PS-Block을 제시한다. PS-Block은 블록 단위로 실행시간을 분석할 수 있는 기반으로 프로그램을 작업 단위로 분리하여 분석할 수 있다. 이를 기반으로 TMO의 소스 코드를 분석하고 블록 단위 실행시간을 예측함으로써 시간 정보 결정의 기준으로 하고 실시간 메소드의 적시성 확인을 쉽게 함으로써 실시간성/신뢰성 향상과 개발 기간을 단축할 수 있다.

본 논문은 2장에서 관련 연구로 실시간 시스템과 소스코드 분석 및 분해 과정에 대해 소개하고 3장에서 TMO 기반의 정적 분석 도구를 위해 제안하는 PS-Block 구조의 설계 내용을 기술한다. 마지막으로 4장에서는 결론과 향후 연구 방향을 제시한다.

## 2 관련연구

### 2.1 TMO

TMO 모델은 적시성 서비스 기능을 디자인 단계에서부터 보장하고, 실시간 시스템이 가지는 시간적인 행동, 메시지에 의한 기능적인 행동에 대한 추상화 등을 지원한다. TMO 모델은 실시간 시스템의 시간적 행동을 추상화하기 위한 SpM(Spontaneous Method), 메시지에 의한 행동을 추상화하기 위한 SvM(Service Method), 그리고 이들이 공유하는 데이터를 세그먼트 단위로 저장하기 위한 ODSS (Object Data Store Segments)등으로 구성되어 있고, 이들을 TMO라는 하나의 객체로 모델링 할 수 있게 해준다. TMO의 시간적 행동은 AAC에서 정의되고, 정의된 주기와 제한시간을 가지고 반복 실행되는 SpM에 의해 SvM이 동작한다. (그림 1)은 TMO객체 모델의 기본 구조를 나타낸다.

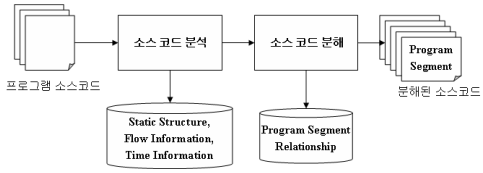



(그림 1) TMO 객체 모델의 기본 구조

TMO는 정시보장 컴퓨팅(Timeliness Guaranteed Computing)을 목표로 제안된 실시간 객체 모델로 경성/연성 실시간 응용뿐만 아니라 일반적인 응용 프로그램에도 사용할 수 있으며, 설계 시 시간 보장 개념을 제공한다[2][3].

### 2.2 소스코드 분석 및 분해기

실시간성/신뢰성 정적 분석 과정에서 실행시간 분석을 위하여 소스의 재구성을 위한 함수의 호출 관계, 명시된 시간 지연 정보, 그리고 중복 계산을 피하기 위한 공유 함수에 대한 정보가 필요하다. 또한, TMO의 SpM과 같이 실시간 프로그램의 특성을 따르는 쓰레드들은 전체 흐름과 상관없이 시간에 의하여 동작하기 때문에 독립적으로 분석이 가능하도록 재구성하고, 재구성된 소스들 사이의 관계 정보를 도출하는 과정이 필요하다. 이러한 과정은 소스코드 분석 및 분해기에서 이루어지며 (그림 2)는 이러한 과정을 나타낸다.



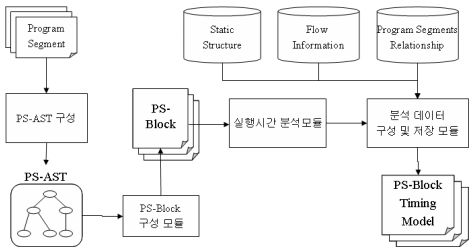

(그림 2) 소스코드 분석 및 분해기

소스코드 분석 과정 중 공유 함수와 전역 변수 등의 정적 구조는 Static Structure를 구성하여 저장하게 되고, 코드에 명시되어 있는 TMO의 AAC와 같은 시간 정보는 Time Information에 저장된다. Flow Information은 소스코드 내부에서 이루어지는 호출 관계와 참조 관계를 가지고, 실행되지 않는 코드를 제거하며, 호출이나 자료 교환이 이루어 지지 않는 코드를 분리한다. 이 Flow Information을 근거로 소스코드 분해 과정에서 소스코드를 Program Segment단위로 분해한다. 여기서 Program Segment란 소스코드를 분석하고 재구축하여 하나의 주기성을 띄고 독립적으로 실행될 수 있는 루틴으로 나눈 것이다. 이와 함께 도출되는 Program Segment Relationship은 Program Segment들 사이의 연관관계를 가진다.

## 3. 본론

실시간 프로그램의 응답 시간을 보장하고, 시간처리의 자유로움을 보장하는 TMO는 설계 단계에서부터 적정한 AAC를 정의해야 한다. 따라서 개발자는 소스코드의 분석을 통해 실행시간을 예측하여 메소드의 시간 처리에 대한 기준점을 제시하고, 실시간성 위반여부를 확인하며, 수정할 수 있도록 실행시간의 분석이 필요하다.

TMO 기반의 정적 분석을 위하여 TMO 프로그램을 분석하면 시간에 의한 행동인 SpM과 메시지에 의한 행동인 SvM에 의해 동작하기 때문에 시간에 따라 동작하는 분리된 흐름들로 구성되는 것을 알 수 있다. 따라서 이 흐름들을 소스코드 분석 및 분해기를 통해 단위 분석을 위한 Program Segment로 나누고 분석을 위한 기초 정보들을 도출한다. 분리된 각각의 흐름인 Program Segment를 시간 계산의 최소 단위로 구문 분석한 PS-AST (Program Segment Abstract Syntax Tree)로 구성하고 각각의 PS-AST 노드를 종류별로 묶어 블록 단위의 실행시간 분석을 위한 기반으로 PS-Block (Program Segment Block)구조로 재구성한다. (그림 3)은 정적 분석 과정을 나타낸다.




(그림 3) 정적 분석 과정

소스코드 분석 및 분해기에서 생성된 Program Segment는 구문 분석을 통해 PS-AST를 생성하고, PS-AST는 블록 단위의 실행시간 분석을 위한 구조인 PS-Block으로 재구축된다. 이를 바탕으로 블록 단위로 실행시간을 분석한다. 분석 데이터 구성 및 저장 모듈에서는 Flow Information과 Static Structure의 분석 결과를 추가하고, Program Segment Relationship을 참조하여 실행시간을 가지는 PS-Block들로 구성된 PS-Block Timing Model을 구성하게 된다.

### 3.1 PS-AST

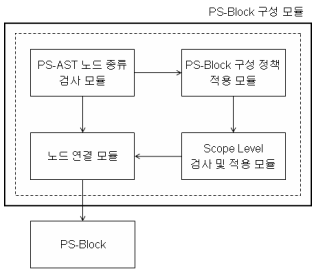
PS-AST는 하나의 주기성을 띄고 독립적으로 실행될 수 있는 Program Segment를 구문 분석하여 생성된 문법 트리로서 실행시간의 분석을 위하여 연산자와 실행문, 변수의 이름을 중심으로 재구성을 거친 구문 트리이다. 재구성은 AST의 노드 중 실행시간 연산에 필요 없는 전역변수를 나타내는 노드와 일반 선언문을 나타내는 노드 그리고, 문법구조에 따라서 생성되어 실행시간 분석과 연관이 없는 비단말 노드들을 트리에서 제거하여 더욱 빠르고 효과적으로 실행시간의 연산이 가능하도록 하는 것이다.

PS-AST의 노드는 각각 하나씩의 연산, 호출, 참조를 나타낸다. 따라서 개별 노드별로 실행시간을 가지고 그 실행시간의 합으로 전체 Program Segment의 실행시간을 나타낼 수 있다.

### 3.2 PS-Block

TMO 프로그램은 SpM에 의해 시간에 따라 분리된 작업으로 구성되어 있어 시간에 따른 작업의 실행시간이 필요하다. 또한, PS-AST는 연산자와 실행문을 중심으로 하는 트리 구조를 이루기 때문에 소스 코드 안의 연산들을 잘 표현할 수 있으나 선택문에 의해 실행되지 않는 경로의 분리가 쉽지 않고 반복문의 반복횟수 계산에 어려움이 따른다. 이에 실행시간 분석을 위한 기반을 마련하기 위해 트리의 노드들을 특정한 기준으로 묶을 필요가 있다.

PS-Block이란 Program Segment를 구분 분석한 PS-AST의 각 노드들을 실행시간의 분석을 위하여 그 종류에 따라 PS-Block 구성 정책을 적용하여 블록 단위로 나누어 재구성한 구조로서 프로그램을 작업 단위로 분리해 분석할 수 있다. 또한, 프로그램의 실행 경로를 명확하게 하고, 반복문의 범위와 구조를 간단하게 표현할 수 있다. (그림 4)는 PS-Block를 구성하는 과정을 나타낸다.




(그림 4) PS-Block 구성 모듈

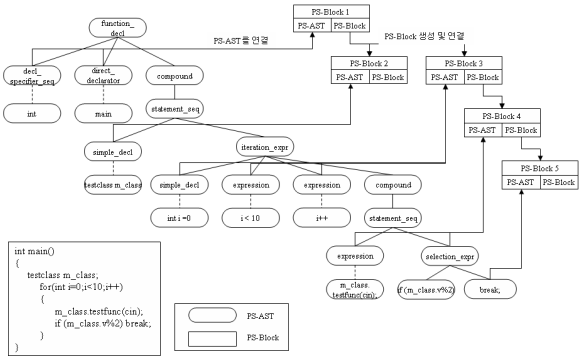
PS-Block은 탐색한 PS-AST 노드의 종류에 따라 구성 정책 적용 모듈에서 블록 생성 및 연결과 통과 여부를 결정한다. PS-AST 노드의 종류별 PS-Block 구성 정책은 <표 1>과 같다.

<표 1> PS-Block 구성 정책

| 노드 종류 | 정책 | 설명 |
|---|---|---|
| function decl | Function 블록 생성 | 함수의 루트 |
| selection expr | Select 블록 생성 | 분기를 하위 블록으로 구분 |
| iteration expr | Iterate 블록 생성 | 조건 블록의 자식으로 내용 블록이 위치 |
| expression | Normal 블록 생성<br>노드 연결 | 실행시간을 구하는 수행문 |
| compound | Pass | |
| simple decl | Pass | 초기화 없을 때 |
| unexpected | Pass | 실행시간 계산이 불필요 |

PS-Block 구성 정책은 메소드 단위 분석을 위한

Function 블록을 생성하고, 선택문의 조건에 따라 일어날 수 있는 분기를 표현하기 위해 Select 블록을 정의하여 실행되는 코드와 실행되지 않는 코드를 구분하며, Iterate 블록을 정의함으로써 반복 실행되는 코드에 대하여 반복 조건을 기준으로 구분해 반복문에 대한 처리를 할 수 있는 기반을 제공한다. Normal 블록은 일반 연산이나 함수 호출 등의 일반적인 실행 코드들의 묶음이고, 다른 블록들을 구성하게 되는 기본적인 블록이 된다. 이는 분기나 반복이 포함되지 않는 단일 코드들의 묶음을 의미한다. Scope Level 검사 및 적용 모듈에서 Scope는 각 PS-AST 노드가 속해야 할 블록을 가리키고, Level은 각 PS-Block이 속하는 부모 블록을 결정하도록 한다. 적용될 정책이 결정되고 검사한 PS-AST 노드, 또는 생성된 PS-Block은 Scope나 Level을 참조해 적절한 PS-Block에 추가된다. (그림 5)는 이와 같은 정책을 적용하여 PS-AST를 PS-Block으로 묶은 구조를 나타낸다.




(그림 5) PS-AST와 PS-Block의 구조

PS-Block은 루틴의 종류에 의하여 그룹화된 구조로 각 분기가 일어나는 지점에서 각각 다른 블록으로 접근할 수 있어 실행 경로의 분석이 용이하고 반복이 일어나는 부분을 하위 블록으로 가지고 있기 때문에 반복문의 구조도 간단히 표현할 수 있다. 또한 각 PS-Block은 블록마다 PS-AST 노드와 실행시간 항목을 가진다. 이 실행시간 항목은 블록에 속한 PS-AST 노드의 실행시간과 하위 블록들의 실행시간의 합으로 나뉜다. 따라서 상위 블록은 하위 블록의 전체 실행시간을 가지고 블록 단위의 개별 실행시간을 분석할 수 있도록 한다.

### 3.3 PS-Block의 특징

PS-Block은 독립된 PS-AST 노드를 가지고 있기 때문에 PS-Block 단위로 독립적으로 실행시간의 분석이 가능하다. 따라서 PS-AST와 PS-Block을 통한 정적 분석은 프로그램의 실행 전에 추상화된 각 메소드들에 대하여 분리된 각 소스 코드의 실행시간을 분석할 수 있게 하고, 최장 경로 분석 방법의 적용이 쉽다. 이처럼 분석 대상을 나눔으로써 각각 실행시간의 분석이 가능하고, 이를 기준점으로 제시하여 개발자의 AAC의 정의와 대상 메소드의 실행시간 적시성 확인을 도울 수 있어 프로그램의 실시간성 및 신뢰성을 향상시키고 개발 기간을 단축할 수 있도록 한다.

## 4. 결론 및 향후 계획

실시간 프로그램은 시간적 정확성을 가져야 하기 때문에 개발자는 이를 고려하여 실행시간을 정의하고 그 정확성 검사를 수행해야 한다. 응답시간을 보장하는 TMO 모델에서도 개발자의 시간 정보 정의와 실시간성/신뢰성의 검증이 필요하고, 이를 위한 정적 분석 도구에 대한 연구가 필요하다.

본 논문에서는 TMO 기반 정적 분석 도구를 위한 실행시간 분석의 기반으로서 PS-AST와 PS-Block 구조를 설계하였다. PS-Block은 프로그램 소스 코드를 블록 단위로 나누어 분석할 수 있도록 함으로써 더욱 자세한 실행시간의 정보를 제공한다. 이를 기준으로 시간 정보의 정의를 돕고 실시간 메소드의 적시성 확인을 쉽게 함으로써 실시간성/신뢰성의 향상과 개발 기간의 단축할 수 있다.

향후 연구 과제로는 본 설계를 바탕으로 정적 분석기를 구현하고 실행시간 분석 결과의 정확성 및 신뢰성 향상을 위하여 운영체제, 통신, 하드웨어 등 실행시간에 영향을 미치는 요인을 분석하여 정적 분석 결과와 실제 실행시간의 차이를 보정하기 위한 연구를 진행할 예정이다.

## 참고문헌

[1] 김태완, 장천현, 김문회, TMO 네트워크로 구성된 분산 실시간 시스템을 위한 실시간성 분석기 설계, ITRC 2004

[2] K. H. (Kane) Kim, A TMO Based Approach to Structuring Real-Time Agents, IEEE, 2002

[3] Kim, K.H. Commanding and reactive control of peripherals in the TMO programming scheme, ISORC 2002

[4] C. Healy, M. Sjödin, V. Rustagi, D. Whalley, Bounding Loop Iterations for Timing Analysis, IEEE, 1998