제26권 제1호(통권 97호) 2013년 3월 ISSN 1738-6322

소프트웨어공학소사이어티 논문지

Journal of Software Engineering Society

VOLUME 26, NUMBER 1, March 2013

사단법인 한국정보과학회 소프트웨어공학소사이어티

Journal of Software Engineering Society

VOLUME 26, NUMBER 1, March 2013

Korean Institute of Information Scientists and Engineers
Software Engineering Society

가변 길이 정보 메시지 최적화 방법†

(An optimization method for variable length information messages)

김진규+ (Jingyu Kim) **강성원**++ (Sungwon Kang) **정필수**+++ (Pilsu Jung) **김정민**++++ (Jungmin Kim) **백하은**+++++ (Haeun Baek)

권구형‡ (Koo Hyung Kwon) **김상수**‡‡ (Sang Soo Kim)

요 약 가변 길이 정보 메시지는 컴퓨터 네트워크 시스템을 통하여 효율적인 정보 제공을 하기 위하여 개발된 통신 프로토콜 표준이다. 이러한 가변 길이 정보 메시지는 정보 수신자의 정보 요구 수준 및 정보 수신자의 정보 접근 수준에 따라 정보의 상세함을 가변적으로 조절 할 수 있도록 설계된 메시지이다. 정보 메시지 최적화 기술은 정보 메시지를 다양한 데이터 압축 기술을 사용하여, 메시지 전체의 물리적인 사이즈를 줄이려고 노력하였다. 정보 메시지 최적화에서는 정보의 정확성을 최우선으로 고려하고 있어, 최적화 전/후가 동일한 비손실 압축 기법을 응용하여 사용하고 있다. 하지만, 이러한 비손실 압축 기법만을 사용하면, 압축효율성이 현저하게 떨어져, 제한된 대역폭을 갖는 무선 네트워크 환경에서의 효율적인 전송이 이루어지지 않는다. 본 논문에서는 가변 길이 정보 메시지를 대상으로, 메시지 필드 단위로 정보의 최적화를 수행하여 메시지의 길이를 물리적으로 좀 더 효과적으로 최적화하도록 시도하였다. 또한, 본 논문에서 제시한 최적화 방법의 효율성을 보이기 위하여, 가변 길이 정보 메시지에 대한 최적화 실험을 수행하였다.

키워드 가변 길이 메시지, 메시지 압축, 메시지 최적화, 메시지 경량화

Abstract Variable length information message is a communication protocol standard in order for computer network systems to provide efficient delivery of information. The variable length information messages were developed for varying and controlling details of information in accordance with message receiver's required information level or information access level. In the previous studies, data compressing techniques have been in use for information message optimization technologies in order to reduce physical sizes of information messages. In optimization technologies for information messages, accuracy of information is considered as the most important factor; therefore, only non-loss compression techniques are applied to the optimization technologies. However, the non-loss compression based information message optimization methods are not efficient in data compression, and these are limited to efficient delivery of information in wireless network environments that have constraint bandwidth. In this paper, we attempt to optimize information in the variable length information messages at message fields in order to reduce physical sizes of messages more efficiently. To demonstrate the efficiency of our approach, we conduct optimization experiments for variable length information messages.

Key words Variable length message, Message compression, Message optimization, Message Lightening

† 본 연구는 국방과학연구소에서 지원하는 '실시간 전술정보 처리를 위한 최적화 방안 연구' 위탁연구과제로 수행되었습니다.

+ 학생회원 : 국방과학연구소
jinlooks@kaist.ac.kr
++ 종신회원 : 한국과학기술원 전산학과 교수
sungwon.kang@kaist.ac.kr
+++ 학생회원 한국과학기술원 전산학과
psjung@kaist.ac.kr
++++ 학생회원 현대오트론
jmin11@kaist.ac.kr
+++++ 학생회원 국방과학연구소
haeun@kaist.ac.kr

‡ 비 회 원 : 국방과학연구소
: koohyung@add.re.kr
‡‡ 비 회 원 : 국방과학연구소
plus@add.re.kr

1. 서 론

가변 길이 정보 메시지는 효율적인 정보 제공을 위하여 다양한 데이터 통신 프로토콜에서 사용되어 왔다. 가변 길이 정보 메시지는 정보 필드와 문법 필드로 구성된다. 정보 필드는 전달하고자 하는 데이터를 표현하는 필드이고, 문법 필드는 존재 지시자(Present Indicator)와 같이 특정 정보 필드에 대하여 문법적인 특징을 제공하는 필드이다. 가변 길이 정보 메시지는 정보 필드에 문법 필드를 조합하여 메시지 명세서 상의 가변적인 길이를 제공한다. 이러한 가변성은 정보 사용자의 정보 요구 수준 및 정보 메시지의 수신자의 정보 접근 수준(Information Access Level)에 따라 정보의 정밀함을 조절하여, 효율적인 데이터 통신을 제공한다. 가변 길이 정보 메시지의 대표적인 예로, IPv6 [9], Variable Message Format (VMF)[1, 2], Asynchronous Transfer Mode (ATM) [4], Re-source Reservation Protocol (RSVP) [12], 그리고 ROHC [10]가 있다.

기존의 정보 메시지 최적화 기술은 정보 메시지를 다양한 데이터 압축 기술을 사용하여, 메시지 전체의 물리적인 사이즈를 줄이려고 노력하였다[5, 6, 7, 8, 11]. 또한 기존의 기술에서는 정보의 정확성을 최우선으로 고려하고 있다. 그로 인하여, 정보 메시지 최적화 수행 시, 압축 전/후가 동일한 비손실 압축 기법(Non-loss Compression)만[5, 6, 7]을 사용하였다. 하지만, 이러한 비손실 압축 기법만을 가변 길이 정보 메시지에 적용하면, 상세한 정보를 표현한 가변 길이 정보 메시지는 물리적인 사이즈가 크고, 압축효율성이 현저하게 떨어져, 제한된 대역폭을 갖는 무선 네트워크 환경에서의 효율적인 전송이 이루어지지 않는다.

본 논문에서는 가변 길이 정보 메시지를 대상으로, 메시지 필드 단위로 최적화를 수행하였다. 본 논문에서는 정보 메시지에 의미적으로는 완결하지만, 정보 손실이 있는 최적화를 수행하여, 메시지의 길이를 물리적으로 좀 더 효과적으로 최적화하도록 시도하였다. 또한, 본 논문에서 제시한 최적화 방법의 효율성을 증명하기 위하여, 가변 길이 정보 메시지에 대한 최적화 실험을 수행하였다.

본 논문의 구성은 다음과 같다. 2절에서는 정보 메시지 최적화에 대한 관련 연구를 논의한다. 3절에서는 가변 길이 정보 메시지에 대하여 논의한다. 4절에서는 가변 길이 정보 메시지의 최적화 방법을 제안한다. 5절에서는 가변 길이 정보 메시지 최적화 실험을 수행하였으며, 6절에서는 본 논문의 기여점에 대하여 논의한다.

2. 관련 연구

서론에서 논의한 것처럼, 기존의 정보 메시지 최적화 기술은 정보 메시지를 다양한 데이터 압축 기술을 사용하여, 메시지 전체의 물리적인 사이즈를 줄이려고 노력하였다[5, 6, 7, 8, 11]. 데이터 압축 기법은 손실 압축(Loss Compression) 기법과 비손실 압축(Non-loss Compr-ession)기법으로 분류된다.

첫 번째로, 손실 압축 기법은 데이터 압축으로 인하여 데이터 손실이 발생하는 압축 기법이다. 주로, 멀티미디어 데이터 같이 내용이 복잡하여 작은 차이들을 인간의 감지 능력으로 구분하기 어려운 정도의 손실만을 줌으로써 데이터 자체에는 큰 손실을 주지 않는 기법이다 [11]. 손실 압축 기법 응용분야는 주로 사진, 동영상, 음악 등의 멀티미디어 데이터이며, 주로 주파수 분석과, 주파수 변환을 바탕으로 압축이 이루어진다.

두 번째로, 비손실 압축 기법은 압축 전/후가 동일하여 압축으로 인해 손실되는 데이터가 전혀 발생하지 않는 압축 기법이다. 비손실 압축 기법의

대상은 주로 문자 혹은 도형 같은 정확성이 요구되는 데이터이다. 비손실 데이터 압축 기법은 다시 (1) Run-Length 압축 기법, (2) Huffman 압축 기법, (3) Lempel-Ziv 압축 기법으로 분류된다 [8].

(1) Run-Length 압축 기법은 가장 단순한 기법으로, 동일한 비트가 연속해서 있을 경우, 그 비트 값과 그 비트 값이 몇 번 반복되는 지 수치로 기록하는 압축원리를 이용한다 [7].

(2) Huffman 압축 기법은 빈도가 높은 바이트는 적은 비트 수로, 빈도가 낮은 바이트는 많은 비트수로 그 표현을 재정의하여 데이터를 압축하는 원리를 이용한다 [6].

(3) Lempel-Ziv 압축 기법은 동적 사전(Dynamic Dictionary)를 이용하는 방법으로, 데이터에 출현하는 단어(Word)들 이진트리(Binary Tree) 혹은 해시를 이용한 검색 구조에 삽입하여 동적 사전을 구현한 다음, 압축하려는 데이터에 동적 사전에 수록되어 있는 단어가 있으며 그에 대한 포인터를 그 단어를 대처하도록 하는 압축 기법이다 [5].

기존의 기술에서는 그동안 정보 메시지의 정확성을 최우선으로 고려하고 있다. 그로 인하여, 정보 메시지 최적화 수행 시, 비손실 압축 기법만을 사용하였다. 하지만, 이러한 비손실 압축 기법만을 사용하면, 압축효율성이 현저하게 떨어져, 제한된 대역폭을 갖는 무선 네트워크 환경에서의 효율적인 전송이 이루어지지 않는다.

가변 길이 정보 메시지는 컴퓨터 네트워크 시스템을 통하여 효율적인 정보 제공을 하기 위하여 개발된 통신 프로토콜 표준이다. 이러한 가변 길이 정보 메시지는 정보 수신자의 정보 요구 수준 및 정보 수신자의 정보 접근 수준에 따라 정보의 상세함을 가변적으로 조절 할 수 있도록 설계된 메시지이다. 하지만, 상세한 정보를 표현한 가변 길이 정보 메시지는 제한된 대역폭을 가지는 무선 통신환경에서 효율적인 정보 교환이 이루어지지 않는다.

본 논문에서는 가변 길이 정보 메시지를 대상으로, 메시지 필드 단위로 최적화를 수행하였다. 본 논문에서는 정보 메시지에 의미적으로는 완결하지만, 정보 손실이 있는 최적화를 수행하여, 향상된 메시지 최적화 결과를 획득하도록 시도하였다.

3. 가변 길이 정보 메시지

가변 길이 정보 메시지는 컴퓨터 네트워크 시스템을 통하여 효율적인 정보 제공을 하기 위하여 개발된 통신 프로토콜 표준이다. 가변 길이 정보 메시지의 예는 다음과 같다: IPv6 [9], Variable Message Format (VMF) [1, 2], Asynchronous Transfer Mode (ATM) [4], Resource Reservation Protocol (RSVP) [12], 그리고 ROHC [10]. 하지만, 이들 가변 길이이 정보 메시지들 중에서 VMF만이 제한된 대역폭을 가지는 무선통신환경에서 구현되고 있어, 메시지 최적화 연구가 수행되어 왔으며, 그 외에 ATM, RSVP, 및 ROHC는 메시지 최적화 연구가 수행되지 않았다.

가변 길이 정보 메시지들은 문법필드와 정보 필드들을 조합하여 메시지의 가변성을 표현한다. 가변 길이 메시지의 문법필드들은 다음과 같다 [1, 2, 3]: FPI (Field Presence Indicator), GPI (Group Presence Indicator), FRI (Field Recurrence Indicator), 그리고 GRI (Group Recurrence Indicator).

FPI (Field Presence Indicator): FPI의 필드 데이터가 '1' 이면, FPI 필드 다음에 연속되는 하나의 정보필드가 존재함을 의미한다. FPI의 필드 데이터가 '0' 이면, FPI 필드 다음에 연속되는 하나의 정보필드가 존재하지 않음을 의미한다.

GPI (Group Presence Indicator): GPI의 필드 데이터가 '1' 이면, GPI 필드 다음에 연속되는 하나의 정보필드 그룹이 존재함을 의미한다. GPI의 필드 데이터가 '0' 이면, GPI 필드 다음에 연속되는 하나의 정보필드 그룹이 존재 하지 않음을 의미한다.

FRI (Field Recurrence Indicator): FRI의 필드 데이터가 '1' 이면, FRI 필드 다음에 연속되는 하나의 이상의 정보필드가 반복되어 존재함을 의미한다. FRI의 필드 데이터가 '0' 이면, FRI 필드 다음에 연속되는 하나의 정보필드가 존재하지만, 반복되어 있지 않음을 의미한다.

GRI (Group Recurrence Indicator): GRI의 필드 데이터가 '1' 이면, GRI 필드 다음에 연속되는 하나의 정보필드 그룹이 반복되어 존재함을 의미한다. GRI의 필드 데이터가 '0' 이면, GRI 필드 다음에 연속되는 하나의 정보필드 그룹이 존재하지만, 반복되어 있지 않음을 의미한다.

가변 길이 정보 메시지들은 다양한 문법필드와 정보필드들을 조합하여 메시지의 정보를 표현하기 때문에, 메시지 정보의 의미적 완결성을 위하여 다음의 의미 규칙을 정의한다 [1, 2, 3].

사례 규칙 (Case Rule): 메시지의 특정 메시지 필드 데이터들이 메시지의 특정 의미 정보 유형을 지정하는 의미 규칙이다. 의미 규칙은 메시지 필드의 필드 데이터들이 의미적으로 조합되어 나타내는 정보를 가리킨다.

조건 규칙 (IF-Then-Else Rule): 의미적으로 메시지의 특정 메시지 필드 데이터가 다른 메시지 필드의 데이터를 제약하는 의미 규칙이다.

조건 규칙은 조건절과 수행절에 따라, 아래와 같이 4가지 유형으로 세분화 될 수 있다:

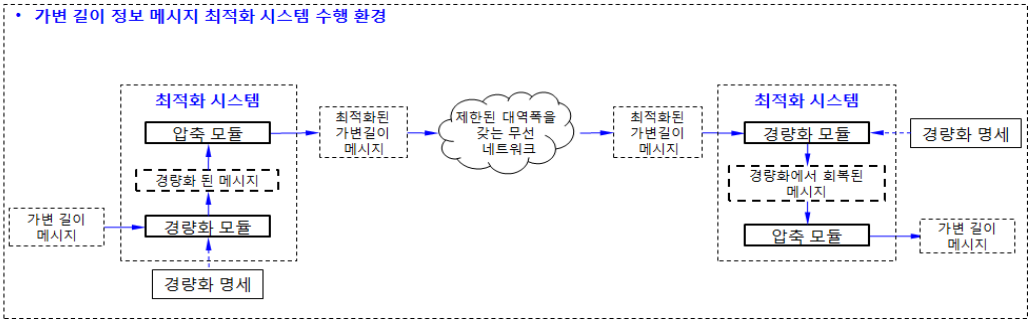
1) 존재 유형(Presence Type):
Ex) IF Field A is presented(='assigned any field data'), THEN Field B is presented
2) 값 할당-존재 유형(Value Assignment- Presence Type):
Ex) IF Field A == 1, THEN Field B is presented
3) 존재-값 할당 유형(Presence-Value Assignment Type):
Ex) IF Field A is presented, THEN Field B == 4
4) 값 할당 유형(Value Assignment Type):
Ex) IF Field A == 1, THEN Field B == 4

본 논문에서는 이러한 가변 길이 정보 메시지들의 특징을 고려하여, 가변 길이 정보 메시지에 대한 최적화 방법을 제안한다.

4. 가변 길이 정보 메시지 최적화 방법

본 절에서는 가변 길이 정보 메시지에 대한 최적화 방법을 제시한다. 가변 길이 정보 메시지 최적화 방법의 배경 지식으로서, 먼저, 가변 길이 정보 메시지에 대한 최적화 방법이 구현된 시스템의 수행 환경을 설명한다.

[그림 1]은 가변 길이 정보 메시지 최적화 시스템의 수행 환경을 나타내고 있다. 가변 길이 정보 메시지 최적화 시스템은 경량화 모듈과 압축 모듈로 구성된다. 가변 길이 메시지는 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈을 통하여 경량화되며, 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈은 경량화된 가변 길이 메시지 명세를 이용하여 경량화를 수행한다.



[그림 1] 가변 길이 메시지 최적화 시스템 수행 환경

경량화된 메시지는 다시 가변 길이 정보 메시지 최적화 시스템의 압축 모듈의 입력이 된다. 경량화된 메시지는 가변 길이 정보 메시지 최적화 시스템의 압축 모듈을 이용하여 각 정보 필드의 데이터가 압축된다. 가변 길이 정보 메시지 최적화 시스템은 압축 모듈을 통하여 압축된 메시지를 최적화된 가변 길이 메시지로 지정하고, 최적화된 가변 길이 메시지는 제한된 대역폭을 가지는 네트워크를 통하여 수신자 측의 가변 길이 정보 메시지 최적화 시스템으로 전달한다.

수신자 측의 가변 길이 정보 메시지 최적화 시스템은 최적화된 가변 길이 메시지를 수신한다. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈은 경량화 명세를 통하여 경량화-회복을 수행한다. 경량화에서 회복된 메시지는 다시 가변 길이 정보 메시지 최적화 시스템의 압축 모듈의 입력이 되며, 경량화에서 회복된 메시지는 가변 길이 정보 메시지 최적화 시스템의 압축 모듈을 통하여, 압축이 수행된 정보 필드에 대하여 압축-회복을 수행한다. 가변 길이 정보 메시지 최적화 시스템은 압축 모듈을 통하여 회복된 가변 길이 메시지를 메시지 수신자에게 전달한다.

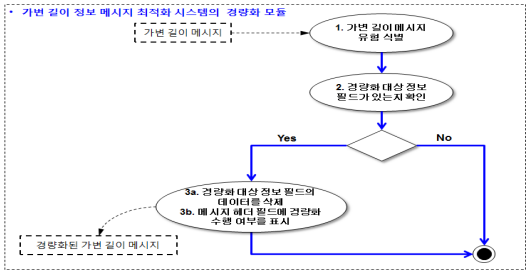
본 논문에서는 가변 길이 정보 메시지 최적화 시스템을 통하여 최적화된 정보 메시지를 생성한다. 우선, 가변 길이 정보 메시지 최적화 시스템은 본 논문에서 제시하는 가변 길이 정보 메시지 최적화 절차 그리고 가변 길이 정보 메시지 최적화-회복 절차에 따라, 최적화 및 최적화-회복을 수행한다.

또한, 가변 길이 정보 메시지 도메인 혹은 프로토콜 전문가는 본 논문에서 제시하는 가변 길이 정보 메시지 경량화 명세 정의 규칙을 바탕으로 경량화된 가변 길이 정보 메시지 명세를 생성한다. 생성된 경량화된 가변 길이 정보 메시지 명세는 가변 길이 정보 메시지 최적화 시스템이 메시지 최적화를 수행할 때 참조할 수 있도록 한다. 또한 가변 길이 정보 메시지 최적화 시스템은 본 논문에서 제시하는 가변 길이 정보 메시지 경량화 변환 규칙 그리고 가변 길이 정보 메시지 경량화 회복 규칙을 참조하여 최적화를 수행한다.

본 논문에서 제시하는 가변 길이 정보 메시지 최적화 방법은 다음과 같이 구성된다: (1) 가변 길이 정보 메시지 최적화 절차, (2) 가변 길이 정보 메시지 최적화-회복 절차, (3) 가변 길이 정보 메시지 경량화 명세 정의 규칙, (4) 가변 길이 정보 메시지 경량화 변환 규칙, 그리고 (5) 가변 길이 정보 메시지 경량화 회복 규칙.

4.1 가변 길이 정보 최적화 절차

가변 길이 정보 메시지 최적화 절차는 아래의 두 단계로 이루어진다:



[그림 2] 최적화를 위한 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈 활동 다이어그램

단계 1) 정보 필드 삭제: 가변 길이 정보 메시지에서 부가적인 정보를 표현하는 정보 필드를 삭제한다.

단계 2) 정보 필드 압축: 가변 길이 정보 메시지 필드들 중에서 압축 가능한 정보 필드 데이터를 식별하고 압축하여 전송한다.

[그림 2]와 같이, 가변 길이 정보 메시지 최적화 시스템은 경량화 모듈을 통하여 다음의 경량화 절차를 진행한다.

1. 가변 길이 정보 메시지 최적화 시스템에 입력된 정보 메시지의 메시지 헤더필드를 확인하여, 어떠한 유형의 정보 메시지인지를 파악한다.
2. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈은 경량화된 가변 길이 정보 메시지 명세를 참조하여 입력된 정보 메시지에서 경량화 대상 정보 필드가 있는지 확인한다.
3. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈은 경량화 대상 정보 필드의 데이터를 삭제하고, 메시지 헤더필드에 경량화 수행 여부를 표시한다.

최종적으로, 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈에서 생성된 경량화된 정보 메시지는 압축 모듈로 전달된다. [그림 3]과 같이, 가변 길이 정보 메시지 최적화 시스템은 압축 모듈을 통하여 다음의 압축 절차를 진행한다.

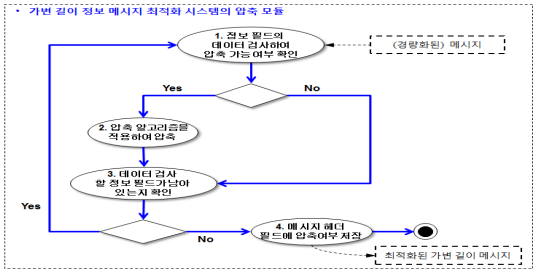
1. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈에서 전달된 정보 메시지는 (문법필드를 제외) 정보필드의 데이터를 검사하여 압축 가능성 여부를 확인한다.
2. 압축 가능한 데이터일 경우, 압축 알고리즘을 적용하여 압축한다.
3. 데이터 검사를 수행할 정보 필드가 남아 있는지 확인한다.
4. 데이터 검사를 수행할 정보 필드가 남아 있으면, 위의 절차들을 반복적으로 수행하고, 더 이상 데이터 검사를 수행할 정보 필드가 남아 있지 않으면, 메시지 헤더필드에 압축 여부를 저장한다. (단, 한번 이상의 압축 알고리즘이 적용 되었을 경우에만, 압축여부를 저장한다)

최종적으로, 가변 길이 정보 메시지 최적화 시스템의 압축 모듈에서 생성된 압축된 정보 메시지는 최적화된 메시지로 지정된다. 최적화된 정보 메시지는 네트워크 망을 통하여 수신자 측에 있는 가변 길이 정보 메시지 최적화 시스템에게 전달된다.

4.2 가변 길이 정보 최적화-회복 절차

가변 길이 정보 메시지 최적화-회복 절차는 아래의 두 단계로 이루어진다.

단계 1) 정보 필드 복원: 가변 길이 정보 메시지에서 삭제된 정보 필드를 경량화 회복 규칙에 따라 회복한다.



[그림 3] 최적화를 위한 가변 길이 정보 메시지 최적화 시스템의 압축 모듈 활동 다이어그램

단계 2) 정보 필드 압축-회복: 가변 길이 정보 메시지 필드들 중에서 압축된 정보 필드 데이터를 식별하고 회복한다.

[그림 4]와 같이, 가변 길이 정보 메시지 최적화 시스템은 경량화 모듈을 통하여 다음의 경량화-회복 절차를 진행한다.

1. 가변 길이 정보 메시지 최적화 시스템에 입력된 (최적화된) 정보 메시지의 헤더필드를 확인하여, 어떠한 유형의 정보 메시지 인지를 파악한다.
2. 입력된 정보 메시지의 헤더필드를 확인하여, 경량화가 수행되었는지 확인한다.
3. 경량화가 수행된 정보 메시지에 대하여, 삭제된 메시지 필드 데이터를 경량화 회복 규칙에 따라 복원한다. 또한, 메시지 헤더 필드에 경량화 수행 여부 표시를 삭제한다.

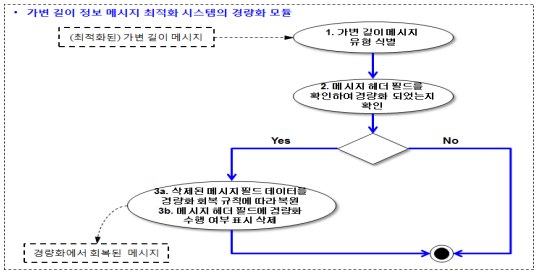
최종적으로, 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈에서 생성된 경량화에서 회복된 정보 메시지는 압축 모듈로 전달된다. [그림 5]와 같이, 가변 길이 정보 메시지 최적화 시스템은 압축 모듈을 통하여 다음과 같은 경량화-회복 절차를 진행한다:

1. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈에서 전달된 정보 메시지의 헤더필드를 보고 압축여부를 확인한다.
2. 압축 알고리즘을 적용한 정보 메시지에 대하여, (문법필드를 제외) 정보필드의 데이터를 검사하여 압축된 정보 필드 인지를 확인한다.
3. 압축된 정보 필드일 경우, 회복 알고리즘을 적용하여 회복한다.
4. 압축 여부 검사를 수행할 정보 필드가 남아 있는지 확인한다.
5. 압축 여부 검사를 수행할 정보 필드가 남아 있으면, 위의 2번과 3번 절차들을 반복적으로 수행하고, 더 이상 압축 여부 검사를 수행할 정보 필드가 남아 있지 않으면, 메시지 헤더필드의 압축여부를 삭제한다.

최종적으로, 가변 길이 정보 메시지 최적화 시스템의 압축 모듈에서 압축 회복된 정보 메시지는 수신자에게 전달된다.

4.3 가변 길이 메시지 경량화 명세 정의 규칙

본 절에서는 가변 길이 정보 메시지 명세에서 경량화 대상이 되는 정보 필드를 선정하는 경량화 명세 정의 규칙들을 제시한다. 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈은 다음의 경량화 명세 정의 규칙으로 바탕으로 생성된 경량화된 가변 길이 정보 메시지 명세를 참조하여 부가적인 정보를 표현하는 정보 필드를 삭제한다.



[그림 4] 최적화-회복을 위한 가변 길이 정보 메시지 최적화 시스템의 경량화 모듈 활동 다이어그램

경량화 명세 정의 규칙-1

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다. (도메인 전문가는 메시지가 사용되는 도메인의 전문가 혹은 메시지에 관련된 프로토콜 전문가를 의미한다)

조건 2) 경량화 대상 정보 필드가 문법 필드에 영향을 받지 않는 독립적인 정보 필드이다.

수행 - 메시지 명세에 해당 정보 필드를 경량화 대상 필드로 표시한다.

경량화 명세 정의 규칙-2

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 해당 경량화 대상 후보 필드가 FPI 혹은 FRI 문법필드에 영향을 받는다.

수행 - 메시지 명세에 해당 정보 필드와 문법 필드 모두를 경량화 대상 필드로 표시한다

경량화 명세 정의 규칙-3

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 해당 경량화 대상 후보 필드가 GPI 혹은 GRI 문법필드에 영향을 받는다.

수행 1 - 해당 문법필드에 영향을 받는 모든 정보 필드가 경량화 대상인 경우, 메시지 명세에 해당 정보 필드들과 문법 필드 모두 경량화 대상 필드로 표시한다.

수행 2 - 해당 문법필드에 영향을 받는 모든 정보 필드가 경량화 대상이 아닌 경우, 메시지 명세에 해당 정보 필드만 경량화 대상 필드로 표시한다.

경량화 명세 정의 규칙-4

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 메시지 사례 규칙에 해당 후보 필드가 명시 되어 있다.

수행 - 메시지 명세에 해당 필드는 경량화 대상 필드로 표시하지 않는다.

부가 설명) 사례 규칙에 명시된 메시지 필드는 전술 정보의 유형을 결정하는 주요한 메시지 필드들이므로 경량화 대상에 포함하지 않는다.

경량화 명세 정의 규칙-5

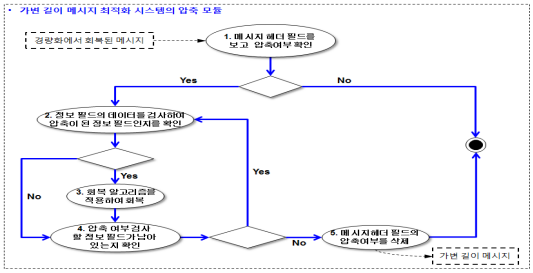
조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 존재 유형의 조건 규칙에서 경량화 대상 후보 필드가 조건절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시한다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시하지 않는다.

부가 설명) 존재 유형의 조건 규칙에서 조건절의 메시지 필드가 존재하면, 수행절의 메시지 필드와 논리적으로 한 메시지에 동시에 존재



[그림 5] 최적화-회복을 위한 가변 길이 정보 메시지 최적화 시스템의 압축 모듈 활동 다이어그램

하여야 한다. 하지만, 조건절의 전술 정보가 삭제되면, 수행절의 메시지 필드도 삭제되어야 한다는 조건은 아니다.

경량화 명세 정의 규칙-6

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 존재 유형의 조건 규칙에서 경량화 대상 후보 필드가 수행절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시하지 않는다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시한다.

부가 설명) 조건절의 전술 데이터가 수행절의 전술 데이터에 영향을 주지만, 수행절의 전술 데이터는 조건절의 전술 데이터에 영향을 주지 않는다.

경량화 명세 정의 규칙-7

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 값 할당-존재 유형의 조건 규칙에서 경량화 대상 후보 필드가 조건절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시한다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시하지 않는다.

부가 설명) 값 할당-존재 유형의 조건 규칙에서 조건절의 메시지 필드의 어떤 특정 전술 데이터에만 영향을 받기 때문에, 수행절 메시지 필드는 영향을 받지 않는다.

경량화 명세 정의 규칙-8

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 값 할당-존재 유형의 조건 규칙에서 경량화 대상 후보 필드가 수행절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시하지 않는다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시한다.

부가 설명) 조건절의 전술 데이터가 수행절의 전술 데이터에 영향을 주지만, 수행절의 전술 데이터는 조건절의 전술 데이터에 영향을 주지 않는다.

경량화 명세 정의 규칙-9

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 존재-값 할당 유형의 조건 규칙에서 경량화 대상 후보 필드가 조건절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시한다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시하지 않는다.

부가 설명) 존재-값 할당 유형의 조건 규칙에서 어떤 메시지 필드와 수행절의 어떤 특정 전술 데이터가 논리적으로 한 메시지에 동시에 존재해야 한다. 조건절의 전술 데이터가 삭제되면, 수행절에서도 메시지 필드가 특정 전술 데이터를 가져야 한다는 조건이 생략되는 것뿐, 삭제되어도 된다는 논리적인 의미를 가지는 것이 아니다.

경량화 명세 정의 규칙-10

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 존재-값 할당 유형의 조건 규칙에서 경량화 대상 후보 필드가 수행절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시하지 않는다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시한다.

설명) 조건절의 전술 데이터가 수행절의 전술 데이터에 영향을 주지만, 수행절의 전술 데이터는 조건절의 전술 데이터에 영향을 주지 않는다.

경량화 명세 정의 규칙-11

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 값 할당 유형의 조건 규칙에서 경량화 대상 후보 필드가 조건절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시한다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시하지 않는다.

부가 설명) 값 할당 유형의 조건 규칙에서 조건절의 특정 전술 데이터와 수행절의 전술 데이터가 논리적으로 한 메시지에 동시에 존재하여야 한다. 조건절의 전술 데이터가 삭제되면, 수행절에서도 메시지 필드가 특정 전술 데이터를 가져야 한다는 조건이 생략되는 것뿐, 삭제되어도 된다는 조건을 의미하는 것은 아니다.

경량화 명세 정의 규칙-12

조건 1) 도메인 전문가가 경량화 대상 후보 필드를 알고 있다.

조건 2) 값 할당 유형(Value Assignment Type)의 조건 규칙에서 경량화 대상 후보 필드가 수행절에 명시되어 있다.

수행 1) 메시지 명세에서 조건절의 필드를 경량화 대상으로 표시하지 않는다.

수행 2) 메시지 명세에서 수행절의 필드를 경량화 대상으로 표시한다.

부가 설명) 조건절의 전술 데이터가 수행절의 전술 데이터에 영향을 주지만, 수행절의 전술 데이터는 조건절의 전술 데이터에 영향을 주지 않는다.

4.4 가변 길이 정보 메시지 경량화 변환 규칙

본 절에서는 경량화가 정의된 가변 길이 정보 메시지 명세를 참조하여 가변 길이 메시지를 경량화된 가변 길이 메시지로 변환하는 규칙들을 제시한다:

경량화 변환 규칙-1

조건 1) 메시지 명세에 특정 정보 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 정보 필드를 포함된다.

수행 – 메시지의 해당 정보 필드의 데이터를 삭제한다.

경량화 변환 규칙-2

조건 1) 메시지 명세에 특정 FPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 FPI 문법 필드를 포함한다.

수행 – 메시지의 해당 FPI 필드의 데이터를 삭제한다.

경량화 변환 규칙-3

조건 1) 메시지 명세에 특정 FRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 FRI 필드를 포함한다.

수행 – 메시지의 해당 FRI 필드의 데이터를 삭제한다.

경량화 변환 규칙-4

조건 1) 메시지 명세에 특정 GPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 GPI 문법 필드를 포함한다.

수행 – 메시지의 해당 GPI 필드의 데이터를 삭제한다.

경량화 변환 규칙-5

조건 1) 메시지 명세에 특정 GRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 GRI 필드를 포함한다.

수행 – 메시지의 해당 GRI 필드의 데이터를 삭제한다.

4.5 가변 길이 정보 메시지 경량화 회복 규칙

본 절에서는 경량화가 정의된 가변 길이 정보 메시지 명세를 참조하여 경량화된 가변 길이 메시지를 원래의 가변 길이 메시지로 회복하는 규칙들을 제시한다:

경량화 회복 규칙-1

조건 1) 메시지 명세에서 특정 (문법 필드에 영향을 받지 않는 독립적인) 정보 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 정보 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 정보 필드가 필수 필드가 아니다.

수행 – 해당 정보 필드의 데이터 회복은 수행하지 않는다.

경량화 회복 규칙-2

조건 1) 메시지 명세에서 특정 (문법 필드에 영향을 받지 않는 독립적인) 정보 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 정보 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 정보 필드가 필수 필드이다.

수행 – 해당 정보 필드의 데이터를 'default field data' 혹은 'undefined field data'로 복원한다.

경량화 회복 규칙-3

조건 1) 메시지 명세에 특정 FPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 FPI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 FPI 필드가 필수 필드가 아니다.

수행 – 해당 FPI 필드의 데이터 회복은 수행하지 않는다.

경량화 회복 규칙-4

조건 1) 메시지 명세에 특정 FPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 FPI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 FPI 필드가 필수 필드이다.

수행 – 해당 FPI 필드의 데이터를 '0'로 복원한다.

경량화 회복 규칙-5

조건 1) 메시지 명세에 특정 FRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 FRI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 FRI 필드가 필수 필드가 아니다.

수행 – 해당 FRI 필드의 데이터 회복은 수행하지 않는다.

경량화 회복 규칙-6

조건 1) 메시지 명세에 특정 FRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 FRI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 FRI 필드가 필수 필드이다.

조건 4) 해당 FRI에 영향을 받는 정보필드가 경량화 수행되었다.

수행 1 – 해당 FRI 필드의 데이터를 '0'로 복원한다.

수행 2 – 해당 정보 필드의 데이터를 'default field data' 혹은 'undefined field data'로 복원한다.

경량화 회복 규칙-7

조건 1) 메시지 명세에 특정 GPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 GPI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 GPI 필드가 필수 필드가 아니다.

수행 – 해당 GPI 필드의 데이터 회복은 수행하지 않는다.

경량화 회복 규칙-8

조건 1) 메시지 명세에 특정 GPI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 GPI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 GPI 필드가 필수 필드이다.

수행 – 해당 GPI 필드의 데이터를 '0'로 복원한다.

경량화 회복 규칙-9

조건 1) 메시지 명세에 특정 GRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 GRI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 GRI 필드가 필수 필드가 아니다.

수행 – 해당 GPI 필드의 데이터 회복은 수행하지 않는다.

경량화 회복 규칙-10

조건 1) 메시지 명세에 특정 GRI 필드가 경량화 대상 필드로 표시되어 있다.

조건 2) 입력된 가변 길이 메시지에 해당 GRI 필드가 경량화 수행되었다.

조건 3) 경량화가 수행된 해당 GRI 필드가 필수 필드이다.

조건 4) 해당 GRI에 영향을 받는 정보필드들이 경량화 수행되었다.

수행 1 – 해당 GRI 필드의 데이터를 '0'로 복원한다.

수행 2 – 해당 정보 필드들의 데이터를 'default field data' 혹은 'undefined field data'들로 복원한다.

5. 가변 길이 정보 메시지 최적화 실험

본 절에서는 가변 길이 정보 메시지 최적화 실험을 수행한다. 본 절의 실험을 통하여 본 논문에서 제안한 가변 길이 정보 메시지 최적화 방법이 유효함을 증명하고자 한다. 본 실험에서는 국방 도메인에서 사용되는 VMF 메시지(비트-패턴 기반의 전술 데이터 메시지)들을 실험 대상으로 하며, 최적화된 가변 길이 정보 메시지에 대한 유효성을 아래의 공식을 통하여 검증한다.

유효성 검증 = (최적화 이전의 가변 길이 메시지 전송시간 × 네트워크 Hop 수) - [(최적화 이후의 가변 길이 메시지 전송시간 × 네트워크 Hop 수) + 인코딩시간 + 디코딩시간]

유효성 검증 공식은 최적화 이전의 전송시간과 최적화 이후의 최적화 수행한 시간과 최적화된 메시지의 전송시간을 합산한 시간을 비교하는 것이다. 즉, 위의 유효성 검증 공식을 적용하여, 양수의 값을 가지면, 최적화가 유효하게 적용되었다고 고려한다. 본 실험에서는 다음과 같이 3가지 유형의 메시지에 대하여 최적화를 수행한다.

(1) 최상의 경우(Best Case: 최대 최적화 효과를 발휘할 것으로 예상되는 메시지를 선정하여 실험을 수행한다)

(2) 최악의 경우(Worst Case: 최소 경량화 효과를 발휘할 것으로 예상되는 메시지를 선정하여 실험을 수행한다)

(3) 산업체 경우(Industrial Case: VMF 메시지 중에서 국방 도메인에서 실제 사용된 메시지에 대하여 실험을 수행한다)

5.1 실험에서 사용한 가변 길이 정보 메시지 유형

본 실험에서는 각 경우에 대하여 다음의 가변 길이 정보 메시지를 대상으로 최적화를 수행한다. (본 논문에서는 보안상의 이유로 다음의 가변 길이 정보 메시지의 구조 및 메시지 필드 데이터들에 대한 상세한 내용은 논의하지 않는다.)

- 최상의 경우: VMF 방공 탄도탄 경보 메시지
  메시지 목적: 방공작전관련 탄도탄 경보 전파를 위한 전술 데이터 메시지
  네트워크 Hop 수 = 3
- 최악의 경우: VMF 방공 경계/공습경보 메시지
  메시지 목적: 방공작전관련 경계/공습경보 전파를 위한 전술 데이터 메시지
  네트워크 Hop 수 = 3
- 산업체 경우: VMF 위치보고 메시지
  메시지 목적: 아군 및 적군의 위치 정보 전파를 위한 전술 데이터 메시지
  네트워크 Hop 수 = 1

5.2 가변 길이 정보 메시지 최적화 실험 결과

본 절에서는 각 경우에 대하여 해당되는 가변 길이 정보 메시지를 최적화를 수행하였으며, 최적화에 필요한 각 세부 활동들에 대한 소요된 시간을 측정하였다. 측정된 세부 활동들은 다음과 같다:

- 경량화 시간: 부가적인 정보를 표현하는 정보필드 확인 및 삭제 시간
- 정보필드 비트-패턴 압축 시간: 압축가능 확인 및 압축 시간
- 정보필드 비트-패턴 압축-회복 시간: 압축한 비트패턴 확인 및 압축회복 시간
- 경량화-회복 시간: 경량화한 정보 필드 데이터 복원 및 원래의 메시지 명세 기준으로 메시지 복원 시간
- 합계: 전체 메시지 최적화 및 최적화 회복 시간 (합계는 개별 측정요소를 합한 수치가 아니라, 전체 프로세스를 측정한 수치이다)

본 실험에서 사용한 정보필드 비트-패턴 압축 알고리즘은 2 Byte 이상의 정보필드에 대하여 동일한 비트-패턴을 그 비트-패턴과 반복 횟수를 기록하는 Run-Length 압축기법[9]을 응용하였다. 최상의 경우에는 2 Byte 이상의 정보필드에 대하여 최대한 많은 비트-패턴들이 반복되도록 정보필드의 데이터를 설정하였으며, 최악의 경우에는 2 Byte 이상의 정보필드에 대하여 반복되는 비트-패턴이 없도록 정보필드의 데이터를 설정하였다.

(1) 최상의 경우

최상의 경우에는 VMF 방공 탄도탄 경보 메시지를 대상으로 최적화를 수행하였다. VMF 방공 탄도탄 경보 메시지는 최적화 이전의 “1,666비트”에서 최적화 수행 이후 “88”비트”로 최적화가 이루어졌다. <표 1>은 VMF 방공 탄도탄 경보 메시지에 대한 최적화 실험결과를 제시하고 있으며, 실험 결과에 대한 최적화 유효성 검사에 있어서도 “+0.986562초”로 최적화의 유효함을 증명하였다.

<표 1> VMF 방공 탄도탄 경보 메시지 최적화 실험결과

측정요소	측정된 시간(초)
경량화 시간	0.000018
정보필드 비트-패턴 압축시간	0.000362
정보필드 비트-패턴 압축-회복 시간	0.000551
경량화 회복 시간	0.000008
합계	0.000938

● 최적화 유효성 검사:

$$(\frac{1,666\,bits}{4,800bps}) \times 3hop - [(\frac{86\,bits}{4,800\,bps}) \times 3hop + 0.000938)] \\ = +0.986562\,seconds$$

(2) 최악의 경우

최악의 경우에는 VMF 방공 경계/공습경보 메시지를 대상으로 최적화를 수행하였다. VMF 방공 경계/공습경보 메시지는 최적화 이전의 "14비트"에서 최적화 수행 이후 "13"비트"로 최적화가 이루어졌다. <표 2>는 VMF 방공 경계/공습경보 메시지에 대한 최적화 실험결과를 제시하고 있으며, 실험 결과에 대한 최적화 유효성 검사에 있어서도 "+0.000603초"로 최적화의 유효함을 증명하였다.

<표 2> VMF 방공 경계/공습경보 메시지 최적화 실험결과

측정요소	측정된 시간(초)
경량화 시간	0.000006
정보필드 비트-패턴 압축시간	0
정보필드 비트-패턴 압축-회복 시간	0
경량화 회복 시간	0.000003
합계	0.000008

● 최적화 유효성 검사:

$$(\frac{14\,bits}{4,800bps}) \times 3hop - [(\frac{13\,bits}{4,800\,bps}) \times 3hop + 0.000008)] \\ = +0.000603\,seconds$$

(3) 산업체 경우

산업체 경우에는 실제로 국방 도메인에서 사용된 VMF 위치보고 메시지를 대상으로 최적화를 수행하였다. VMF 위치보고 메시지는 최적화 이전의 "261비트"에서 최적화 수행 이후 "205"비트"로 최적화가 이루어졌다. <표 3>은 VMF 위치보고 메시지에 대한 최적화 실험결과를 제시하고 있으며, 실험 결과에 대한 최적화 유효성 검사에 있어서도 "+0.011494초"로 최적화의 유효함을 증명하였다.

<표 3> VMF 위치보고 메시지 최적화 실험 결과

측정요소	측정된 시간(초)
경량화 시간	0.000018
정보필드 비트-패턴 압축시간	0.000028
정보필드 비트-패턴 압축-회복 시간	0.000117
경량화 회복 시간	0.000010
합계	0.000172

● 최적화 유효성 검사:

$$(\frac{261\,bits}{4,800bps}) \times 1hop - [(\frac{205\,bits}{4,800\,bps}) \times 3hop + 0.000172)] \\ = +0.011494\,seconds$$

가변 길이 정보 메시지 최적화 실험에서는 본 절에서 제시한 모든 경우에 대하여 수행한 최적화가 유효함을 증명하였다.

6. 결 론

본 논문은 제한된 대역폭을 갖는 무선 네트워크 환경에서 비효율적인 전송을 해결하기 위한 방법을 제시하였다. 기존의 정보 메시지 최적화 기술은 정보 메시지를 다양한 데이터 압축 기술을 사용하여, 메시지 전체의 물리적인 사이즈를 줄이려고 노력하였다. 기존 기술에서는 정보의 정확성을 최우선으로 고려하였고, 정보 메시지 최적화 수행 시, 압축 전/후가 동일한 비손실 압축 기법만을 사용하였다. 하지만, 이러한 비손실 압축 기법만을 사용하면, 압축효율성이 현저하게 떨어져, 제한된 대역폭을 갖는 무선 네트워크 환경에서의 효율적인 전송이 이루어지지 않는다.

가변 길이 정보 메시지는 컴퓨터 네트워크 시스템을 통하여 효율적인 정보 제공을 하기 위하여 개발된 통신 프로토콜 표준이다. 이러한 가변 길이 정보 메시지는 정보 수신자의 정보 요구 수준에 따라 혹은 정보 수신자의 정보 보안 수준에 따라 정보의 상세함을 가변적으로 조절할 수 있도록 설계된 메시지이다. 하지만, 상세한 정보를 표현하는 가변 길이 정보 메시지는 제한된 대역폭을 가지는 무선 통신환경에서 효율적인 정보교환이 이루어지지 않는다.

본 논문에서는 가변 길이 정보 메시지를 대상으로 하여 메시지 필드 단위의 정보 메시지 최적화 방법을 제시하였다. 본 논문에서는 국방 도메인에서 사용되는 가변 길이 정보 메시지인 VMF 전술 데이터 메시지를 대상으로 메시지 최적화 실험을 수행하였으며, VMF 전술 데이터 메시지 최적화 실험을 통하여 본 논문에서 제안한 최적화 방법의 유효함을 증명하였다. 추후 연구에서 본 연구진은 관련연구에서 언급한 비손실압축 기법을 사용하는 최적화 방법과 본 연구에서 제안한 방법의 비교 실험을 수행할 예정이다.

참 고 문 헌

[1] DoD (Department of Defense), Interface Standard TADIL-J MIL STD 6016A, JEBCA Teston VA, April 1999.

[2] DoD (Department of Defense), MIL-STD-6017, Variable Message Format(VMF), 2002.

[3] Go, K., Kang, Sungwon, Kim, M., Lee, J., “A Systematic Test Case Generation Approach for Testing Message Length Variability”, IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST), pp.397~406, 2011.

[4] Handel, R., Huber, M. N., Schroder, S., ATM Networks: Concepts, Protocols, Applications, Addison-Wesley Longman Ltd., Essex, UK, 1998.

[5] 진용선, 정정화, “실시간 데이터 압축을 위한 Lempel-Ziv 압축기의 효과적인 구조의 제안”, 전자공학논문지, 제37권, 제3호, 2000.

[6] 김현철, 류재경, 정선이, 정진욱, “허프만 부호를 이용한 압축/암호 결합 알고리즘”, 한국정보과학회 학술발표논문집, 제18권, 제1호, 1991.

[7] 박일남, “전자상거래 문서 보안을 위한 Run- length 거리 혼합 합성 알고리즘에 관한 연구”, 한국정보기술학회논문지, 제4권, 제4호, 2006.

[8] Sayood, K., “Introduction to Data Compression,” 2nd Ed., Morgan Kaufmann Publishers, 2000.

[9] The Internet Society, RFC2460: Internet Protocol, Version 6 (IPv6) Specification, 1998.

[10] The Internet Society, RFC3095: Robust Header Compression (ROHC) Framework and Four Profiles, 2001.

[11] Tomohiko, U., “Introduction to the document data compression algorithm,” Sungandang, 1995.

[12] Zhang, L., Deering, S., Estrin, D, Shenker, S., and Zappala, D., “RSVP: A New Resource ReSerVation Protocol,” IEEE Network, pp.8~18, 1993.

저자 소개

김 진 규

2013년 KAIST 졸업(박사)
2013년~현재 국방과학연구소 근무 중

김 정 민

2013년 KAIST 졸업(석사)
2013년~현재 현대오트론 근무 중

강 성 원

1982 서울대학교 사회과학대학 졸업
1989 University of Iowa 전산학 석사
1992 University of Iowa 전산학 박사
1993~2001한국통신(KT) 선임연구원
1995~1996 미국 국립표준기술연구소(NIST) 객원연구원
2001~2005 한국정보통신대학교 조교수
2003~현재 카네기멜론 대학교 MSE 프로그램 겸임교수
2005~2009 한국정보통신대학교 부교수
2009~현재 KAIST 전산학과 부교수

백 하 은

2013년 KAIST 졸업(석사)
2013년~현재 국방과학연구소 근무 중

정 필 수

2012년 충남대학교 졸업(학사)
2012년~현재 KAIST 전산학과 석사과정 재학 중

권 구 형

2003년 고려대학교 졸업(석사)
2006년~현재 국방과학연구소 근무 중

김 상 수

2003년 경북대학교 졸업(석사)
2003년~현재 국방과학연구소 근무 중

아키텍처 변환 패턴을 이용한 소프트웨어 시스템 진화 프레임워크†

(A Framework for Software System Evolution using Architectural Transformation Pattern)

박태현‡ (Taehyun Park) 안 휘‡ (Hwi Ahn) 강성원¶ (Sungwon Kang)
박종빈¥ (Jongbin Park) 황상철¥ (Sangcheol Hwang)

요 약 소프트웨어 시스템 진화는 소프트웨어 시스템의 유지보수의 일종으로 계획적이고 체계적인 유지보수 프로세스이다. 소프트웨어 진화 연구는 기존 시스템의 비용편익 분석을 통해 시스템의 유지 가치에 대한 판단 근거를 제공하며, 아키텍처를 기반으로 하는 진화는 반복적인 진화 작업의 자동화를 가능케 하여 유지보수 비용 감소를 가능하게 해주는 연구이다. 본 논문에서는 아키텍처 변환 패턴을 이용한 소프트웨어 시스템 진화 프레임워크를 제안한다.

키워드 아키텍처, 소프트웨어 시스템 진화 프레임워크, 아키텍처 변환 패턴

Abstract Software System Evolution is more planned and systematic maintenance process as well as a type of maintenance. The research of software evolution provides basis of decisions for maintenance value through cost-benefit analysis of legacy system and architecture-based software evolution enables engineers to reduce maintenance cost by automation of repetitive evolution tasks. This paper proposes a framework for software system evolution using architectural transformation pattern.

Key words Architecture, Software System Evolution Framework, Architecture Transformation Pattern

1. 서 론

소프트웨어 시스템을 이용하는 유저들의 트렌드와 요구사항들은 지속적이고 빠르게 변화하며, 회사는 생존을 위해 이러한 변화에 빠르게 대응하는 것이 중요하다 [1][2][6]. 그러나 이러한 변화에 매번 새로운 소프트웨어 시스템을 개발하여 대응하는 것은 개발 예산과 시간을 고려하였을 때 불가능하다. 소프트웨어 시스템 진화(Evolution)는 소프트웨어 시스템의 유지보수의 일종으로 계획적이고 체계적인 유지보수 프로세스이다[3]. 소프트웨어 진화 연구는 기존 시스템의 비용편익 분석을 통해 시스템의 유지 가치에 대한 판단 근거를 제공하며, 아키텍처의 변경 없이도 계획된 변화의 범위 안에서 저렴한 비용으로

† 본 연구는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2012-0007069).
‡ 학생회원 : 한국과학기술원 전산학과
〔tpark, ahnhwi〕 @kaist.ac.kr
¶ 종신회원 : 한국과학기술원 전산학과 부교수
sungwon.kang@kaist.ac.kr
¥ 비 회 원 : SK Planet Software Quality Engineering 팀
jongbin@sk.com, k16wire@gmail.com

진화가 가능한 장점을 갖는다. 그렇기 때문에, 기존의 소프트웨어 시스템을 진화시키는 기술은 이러한 변화들에 대응하기 위해 필수적이다 [1][2][4][5].

본 논문에서는 소프트웨어 시스템의 진화 과정의 자동화를 가능하게 해줄 수 있는 아키텍처 변환 패턴을 이용한 소프트웨어 시스템 진화 프레임워크를 제안한다. 아키텍처 변환 패턴을 사용함으로써 회사마다 또는 도메인에 따라 다르게 표현될 수 있는 소프트웨어 시스템 아키텍처와 각 시스템의 진화 방식을 쉽고 효율적으로 표현할 수 있다.

아키텍처 기반의 소프트웨어 기술은 정형화된 아키텍처 모델을 통해 소프트웨어 시스템을 관리, 분석 및 진화를 가능케 해주고, 반복적인 진화 작업의 자동화를 달성할 수 있게 해준다. 이를 통해 소프트웨어 시스템 진화를 체계적으로 진행할 수 있으며, 소프트웨어 시스템의 유지보수 비용 감소를 기대할 수 있다.

또한 본 논문에서는 기존의 아키텍처 기반의 소프트웨어 시스템 진화 기술과 그리고 아키텍처 변환 기술에 대해 고찰하고, 제안하는 소프트웨어 진화 프레임워크를 지원하는 툴로서 ASES(Automatic Service Evolution System)를 소개함으로써 진화 프레임워크를 설명하고 이를 프로토타입 형태로 제공한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 본 논문에서 제안하는 아키텍처 기반의 소프트웨어 시스템 진화의 관련 연구로서, CMU에서 제안한 자가 적응형 소프트웨어 프레임워크를 제안한 Rainbow(레인보우) 프레임워크, SEI에서 제안한 Simplex 아키텍처 연구를 소개한다. 3장에서는 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크 제안에 필요한 소프트웨어 아키텍처 표현법, 아키텍처 표현 언어(ADL, Architecture Description Language), 그리고 아키텍처 변환 패턴 구현에 이용된 아키텍처 변환 명령어들을 제안한 Transformational Architecture Design(TAD)에 대해서 논의한다. 4장에서는 본 논문에서 제안하는 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크를 설명한다. 5장에서는 4장에서 제시한 소프트웨어 시스템 진화 프레임워크를 활용한 사례 연구를 프로토타입 형태로 소개하며, 6장에서 결론 및 향후 연구를 제시한다.

2. 관련 연구

이 절에서는 본 연구에서 제안하는 아키텍처 변환 패턴을 이용한 소프트웨어 시스템 진화 프레임워크와 관련된 기존 연구들을 소개하고, 연구들이 갖는 문제점과 한계에 대해서 대해서 고찰한다.

기존의 아키텍처 기반의 소프트웨어 진화 관련 연구로는 90년대 후반에 Oreizy et al[4][5]이 소프트웨어 아키텍처 모델에 런타임상의 변화를 반영하는 프로세스를 제시하였고 이를 기반으로 K-Component Architecture Meta-model[13], Model-based development[14], Object-oriented design adaptation[15] 등 많은 연구가 진행되었다.

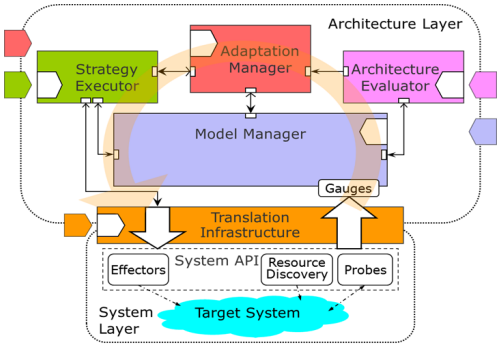
본 논문에서는 그 중에 Rainbow Framework가 제시하는 진화 프로세스 구조와 Simplex Architecture에서 아키텍처 요소들이 진화하는 자세한 프로세스에 대해서 고찰한다.

2.1 Rainbow Framework(레인보우)

Rainbow Framework[1]는 2004년 카네기 멜론 대학교의 D. Garlan과 S. Cheng이 제안한 프레임워크로 추상화된 아키텍처 모델을 사용하여 현재 동작중인 시스템의 런타임 속성을 모니터링 하고 모델을 평가하여 문제가 발생하였을 때 동작 중인 시스템에 적절한 적응을 수행할 수 있도록 한다. 현재 Rainbow, RAIDE[2], Stitch[12] 등 런타임 상에서의 소프트웨어 시스템의 아키텍처 기반의

자가적응 연구들이 카네기 멜론 대학교의 ABLE Project를 통해 활발히 진행되고 있다.

Rainbow 프레임워크의 기능은 다음과 같다. [그림 1]에서 보여주듯이, 소프트웨어 시스템 및 환경의 상태를 모니터링 해주는 모니터링 파트는 다양한 종류의 Probe로 구성되어 있다. 그리고 Gauges는 Model Manager에서 관리가 되는 아키텍처 모델의 속성에 따라 관찰을 하는 부분이다. 새로운 업데이트, 즉 변화(Change)가 발생 했을 시, Architecture Evaluator가 아키텍처 모델에서 정의된 허용치 안에서 시스템이 제대로 구동되고 있는지 확인한다. 만약에 해당 시스템이 정의된 허용치(e.g. 성능) 안에서 적절히 구동되지 못하고 있다면, Adaptation Manager가 정의된 적응 작업을 결정하도록 반응한다. 그리고 마지막으로 Executor가 선택된 자가 적응 작업들을 시스템 레벨의 Effector를 통해 런타임중인 시스템에 적용하여 변화에 대해 대응하여 시스템이 안정된 상태로 구동한다.



[그림 1] Rainbow 자가 적응 프레임워크[1]

Rainbow 프레임워크가 런타임상의 시스템에서 발생하는 변화에 대해 아키텍처 관점에서의 적응 방법을 잘 제시하고 있다. 그러나 Rainbow 프레임워크는 제시하고 있는 구조와 각 요소들의 역할에 맞춰 시스템의 아키텍처 요소가 연결되어야 한다. 그러므로 시스템의 설계 및 개발 과정에서부터 이 점을 고려하여 시스템에 적용되어 있지 않다면 프레임워크에서 제시하는 아키텍처 기반의 자가 적응 기능 발휘가 어려울 수 있다. 예를 들어, 기존에 운영되고 있는 기존 시스템의 아키텍처 대한 주의 깊은 고려 없는 Rainbow 프레임워크 적용은 다양한 문제를 발생시킬 수 있다. 또한 시스템의 변화가 감지되어 적응 전략(Adaptation Strategy)의 형태와 이 전략을 Effector를 통해 시스템에 적용할 때 어떤 형태로 적용하는지에 대한 자세한 설명이 부족하여 실제적으로 적용하기에 어려움이 있다.

2.2 Simplex Architecture

Simplex Architecture는 지속적으로 작동하는 소프트웨어 시스템에서 장애 허용(Fault tolerance) 개념과 작동 중 하드웨어와 소프트웨어 컴포넌트들의 동적 상태에서의 업그레이드(Dynamic upgrade)를 가능하게 하는 프레임워크로써, 1995년 카네기 멜론 대학교의 Software Engineering Institute(SEI)에서 개발되었다[9]. Simplex는 주로 임베디드 시스템에 적용되어 테스트 되었으며, 2000년 초반까지 연구되다가 최근에는 연구 진행이 되지 않고 있는 상태이다.

Simplex Architecture는 업그레이드를 하고자 하는 소프트웨어(또는 하드웨어) 시스템을 컴포넌트와 커넥터의 관점에서 인식한다. Simplex Architecture에서 컴포넌트는 교체 단위(Replacement Unit)를 의미하며, 이는 정형화된 업그레이드 트랜잭션(transaction)을 통해 온라인으로 추가, 삭제, 통합이 가능한 런타임 컴포넌트를 뜻한다. 커넥터는 업그레이드 트랜잭션을 의미한다.

Simplex Architecture의 구조는 계층적으로 되어 있다. System Configuration Manager(SCM)은 다수의 Processor Configuration Manager(PCM)을 포함하고 있으며, 각 PCM은 Sub-system Module들을 가지고 있다. Sub-system Module들은 교체

단위인 Application unit들을 포함하고 있는데, Application unit들은 해당 소프트웨어 시스템의 서비스를 제공하는 단위이다. 이 외에 장애 허용 기능을 수행하는 Safety unit과 이들을 관리하는 Module Management Unit이 Sub-system Module에 속해 있으며, 이들은 PCM을 통해 교체가 가능하다.

Simplex Architecture에서 커넥터의 역할을 하는 업그레이드 트랜잭션은 교체 트랜잭션들로 구성되어 있다. 예를 들어, 하나의 교체 단위를 교체하기 위한 기본적인 교체 트랜젝션은 다음과 같은 순서로 진행된다.

새로운 교체 단위가 생성되고 대기 상태에 있는다.

새로운 입력 값들이 새로운 교체 단위에 제공이 된다. 이 새로운 교체 단위가 출력하는 출력 값들은 에러 유무에 대하여 면밀히 검토된다. 출력 값들은 검토만 되고 실제로 사용되지 않는다.

Module Management Unit은 검토되는 출력 값들이 문제가 없는지 확인하고 문제가 없고 안정적인 상태에 이르렀는지 확인한다.

과거의 교체 단위가 폐기된다.

Simplex Architecture는 계층적인 구조로 이뤄진 소프트웨어 및 하드웨어의 아키텍처 컴포넌트 교체방식을 기반으로 한 진화 방법을 소개하고 있다. 하드웨어와 소프트웨어를 같은 개념에서 다룸으로써 직관적인 형태로 교체 트래젝션과 같이 소프트웨어 진화 방법을 상세히 설명하고 있다. 그러나 Simplex Architecture는 회사나 소프트웨어 도메인에 따라 다르게 존재할 수 있는 진화 방식을 표현하기 힘들다.

3. 배경 지식

이 절에서는 본 연구에서 제안하는 아키텍처 변환 패턴을 이용한 소프트웨어 시스템 진화 프레임워크를 구성 요소의 기본이 되는 소프트웨어 아키텍처 표현 방법과 아키텍처 변환 패턴인 Transformational Architecture Design(TAD)를 설명한다.

3.1 소프트웨어 아키텍처

소프트웨어 아키텍처는 진화하고자 하는 대상이 되는 소프트웨어 시스템과 진화의 목표가 되는 소프트웨어 시스템을 논리적 수준에서 표현한다. 이는 진화의 출발점과 진화의 종착점을 물리적 수준(코드, 실제 파일들)에서 논리적 수준(아키텍처)으로 추상화시켜 준다. 이를 통해 변환 패턴이 소프트웨어 시스템 진화를 위해 적용될 수 있는 기반을 마련해준다.

소프트웨어 아키텍처는 물리적으로 보이거나 만져지지 않는 특성 때문에, 왜곡 없이 정확하게 표현하려면, 다양한 뷰포인트를 통하여 표현하여야 한다.

P. Kruchten의 4+1 아키텍처 뷰포인트 프레임워크[17]를 통해 논리 뷰(Logical view), 실행 뷰(Process view), 개발 뷰(Development view), 물리 뷰(Physical view), 그리고 시나리오(Scenario view)로 분류하였다. 각 뷰들의 간단한 정의는 아래와 같다.

- 논리 뷰: 기능과 요구사항 위주로 소프트웨어 아키텍처를 표현한다.
- 실행 뷰: 소프트웨어 시스템의 실행 시 모습을 표현한다.
- 개발 뷰: 소프트웨어 시스템을 구성하는 컴포넌트들, 또는 서브시스템들의 계층 관계를 표현한다.
- 물리 뷰: 소프트웨어가 배치된 물리적 환경과의 관계를 표현한다.
- 시나리오 뷰: 위의 4개의 뷰들이 하나의 통일된 소프트웨어 시스템을 구성하도록 한다.

SEI의 L. Bass외 연구진은 소프트웨어 아키텍처를 아래와 같은 3개의 아키텍처 뷰로 표현하도록 정의하였다[7].

- 모듈 뷰(Module view): 코드 측면에서 표현한 소프트웨어 아키텍처 뷰로써, 작은 모듈들의 기능적 측면을 고려하여 표현한다.
- 컴포넌트 앤 커넥터 뷰(Component and Connector view): 소프트웨어 시스템의 실행 시 모습을 표현한다.
- 배치 뷰(Allocation view): 소프트웨어가 실행되는 물리적 환경(하드웨어)을 표현한다.

각 아키텍처 뷰들은 소프트웨어 시스템의 서비스를 제공해주는 컴포넌트와 이들을 연결하는 커넥터, 그리고 여러 컴포넌트들과 커넥터가 연결된 형상(Configuration)으로 표현될 수 있다. 예를 들어, L. Bass 등이 제시한 모듈 뷰에서는 기능 모듈이 컴포넌트가 되고, 그들 간의 계층 관계가 커넥터가 된다.

아키텍처를 표현하기 위해서는 일반적인 문서 작업 도구(마이크로소프트 워드, 파워포인트, 한글 등)를 이용하여 컴포넌트, 커넥터를 그릴 수 있다. 또한, UML과 같이 소프트웨어 개발에 일반적으로 사용되는 산출물 표현 언어를 사용하여 아키텍처를 정의할 수도 있다. 하지만 좀 더 상세한 아키텍처 정의를 위해서는 미리 엄격하게 정의된 아키텍처 표현 언어인 ADL을 쓸 수 있다. ADL에는 Acme[4], eXtensible Architecture Description Language(xADL)[8] 등이 있다. 본 연구에서는 카네기 멜론 대학교에서 개발된 Acme를 사용하였다.

Acme는 컴포넌트(Component), 커넥터(Connector), 포트(port), 역할(role)을 기본 아키텍처 구성요소로 가진다. 컴포넌트, 커넥터, 포트의 정의는 앞서 설명된 일반적인 정의와 같다. 역할은 커넥터에 속하는 아키텍처 구성요소로써, 컴포넌트의 포트와 연결되는 지점을 의미한다. Acme는 이클립스 기반의 전용 툴이 존재하며, 이를 통해 시각적인 기호와 텍스트 기반의 기호를 모두 제공한다. 또한 Acme를 사용하여 개발하기 위한 자바 기반의 라이브러리를 제공한다.

3.2 Transformational Architecture Design(TAD)

S. Kang[10]은 소프트웨어 아키텍처 설계를 위한 변환명령어를 제시한다. 기존의 소프트웨어 아키텍처 디자인에서는 먼저 아키텍처부터 설계된 후에 문법적인 유효성(syntactic validity)이 검정된다. 반면에 Transformational Architecture Design (TAD) 방법에서는 유효(valid)한 아키텍처가 설계될 수 있도록 정확하게 정의된 변환 명령어들(transportation operations)이 함께 설계된다. 또한 이 변환명령어들은 자동화된 아키텍처 디자인도 가능케 한다. 하지만 아키텍트는 문법적(Syntactic)으로 유효한 아키텍처뿐만 아니라 의도한 의미(Semantic)까지 정확히 반영하는 아키텍처 설계를 필요로 한다.

소프트웨어 아키텍처 설계가 문법적으로 유효하고 의도한 의미까지 반영되도록 하는 TAD 방법에서는 소프트웨어 아키텍처 디자인에 대한 변환 명령어들과 시스템을 구성하는 컴포넌트가 아키텍처에서 의도된 서비스 제공 여부를 체크하는 알고리즘을 함께 제안한다.

TAD 방식은 다음의 특성을 가진다[10].

- 변환 규칙(transformation rule)들은 정확히 정의된 구문적인 규칙을 가진다.
- 변환 규칙들은 아키텍트가 의도한 의미(Semantic)를 유지할 수도 있으며 문법적인 내용만을 가질 수 있다.
- TAD는 시스템의 기능성(functionality)들을 알게 해주기 때문에 아키텍처 디자인에 대해서 유용한 방법일 뿐만 아니라 어떤 품질을 변경하거나 새로운 기능을 추가하는 시스템의 진화에도 쉽게 수행 될 수 있으며 검증될 수 있다.
- 아키텍처 설계에 의도된 의미(Semantic)는 변환 규칙들이 연속성을 띈 형태인 변환(Transformation)으로 반영 또는 유지시킬 수 있다.
- TAD는 서비스 semantic perspective로부터 변환 명령어들의 의미들을 정의하고 시스템과 컴포넌트들에 대한 서비스를 수행하는 서비스 수행 알고리즘을 활용하여 아키텍트로 하여금 아키텍처 디자인이 그들의 의도에 맞는지 체크할 수 있도록 한다.

소프트웨어 진화를 정의할 수 있는 변환 패턴은 여러 변환 명령어들로 구성이 되어 있는데, 이 명령어들은 크게 아래와 같이 7가지로 분류된다 [10].

1) 생성과 제거(Creation and Destruction): 새로운, 또는 존재하는 컴포넌트, 커넥터, 또는 포트를 생성, 또는 제거한다.
2) 추가(Addition): 새로운 컴포넌트, 혹은 커넥터를 추가한다.
3) 삭제(Deletion): 존재하는 컴포넌트, 혹은 커넥터를 삭제한다.
4) 연결(Connecting): 컴포넌트의 포트와 커넥터의 포트를 연결한다.
5) 단절(Disconnecting): 컴포넌트의 포트와 커넥터의 포트를 끊는다.
6) 분해(Decomposition): 존재하는 컴포넌트를 두 개의 컴포넌트로 분해한다.
7) 결합(Merging): 존재하는 두 개의 컴포넌트를 하나의 컴포넌트로 결합한다.

각각의 분류에는 여러 상세한 명령어들이 속하고 있으며, [그림 2]은 그 중 '추가'에 해당하는 명령어의 한 예이다. 그 외의 상세한 명령어들은 [10]에 정의되어 있다.

add(Σ, Comp) :
대상이 되는 소프트웨어 아키텍처 전체(Σ)에 컴포넌트 Comp를 추가한다.

[그림 2] 컴포넌트를 추가하는 변환 명령어의 예

각 변환 명령어들은 필요한 입력 값들을 갖는다. 예를 들어, [그림 2]에 정의된 add 변환 명령어는 대상이 되는 소프트웨어 아키텍처 전체(Σ)와 추가하려는 컴포넌트 Comp를 입력 값으로 가진다. 이러한 입력 값들은 소프트웨어 아키텍처 표현 요소들 모두가 될 수 있으며, [10]에서는 아래와 같이 분류한다.

1) 시스템(System): 소프트웨어 시스템을 의미한다. 주로 Σ로 표현된다.
2) 컴포넌트: 컴포넌트는 시스템 형상(System configuration)의 가장 기본적인 단위이자, 서비스를 제공하는 기본 단위를 의미한다.
3) 커넥터: 컴포넌트들 간의 서비스를 전달해주는 아키텍처 구성 요소를 의미한다.
4) 컴포넌트와 커넥터의 연결: 시스템에서 컴포넌트는 서비스의 제공자와 사용자로 구분되며, 이들은 반드시 커넥터를 통해 연결되어야 한다.
5) 포트: 컴포넌트와 커넥터가 연결되는 지점을 의미한다.
6) 구조(Structure): 시스템 자신을 포함하여, 분해 가능한(non-atomic) 형상(Configuration)을 의미한다.

이러한 아키텍처 표현 요소들은 다양한 방법으로 표현될 수 있지만, TAD의 변환 패턴에서 정의하는 영역은 아니며 Architecture Description Language (ADL)에서 위의 요소들을 소프트웨어 아키텍처로써 표현하는 방법을 정의하고 있다 [11][16].

4. 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크

4절에서는 아키텍처를 기반으로 한 소프트웨어 시스템 진화 프레임워크를 제시한다. 이를 위하여 4.1절에서는 본 프레임워크를 구성하는 소프트웨어 진화 요소들에 대하여 설명하며, 변환 패턴, 소프트웨어 아키텍처 표현 그리고 소프트웨어 진화 과정(Process)이 설명된다. 4.2절에서는 4.1절에서 설명된 소프트웨어 진화 요소들을 이용한 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크가 제시된다.

4.1 프레임워크 구성요소

아키텍처 기반의 소프트웨어 시스템 진화는 진화의 대상이 되는 소프트웨어와 진화 후의 소프트웨어를 표현하는 아키텍처, 진화 과정을 표현하는 변환 패턴, 그리고 소프트웨어 아키텍처에

변환 패턴이 적용되는 소프트웨어 진화 과정으로 설명 될 수 있다. 그렇기 때문에, 변환 패턴, 소프트웨어 아키텍처, 그리고 소프트웨어 진화 과정을 소프트웨어 진화 프레임워크의 구성 요소들이라 할 수 있다. 이 절에서는 위의 각 구성 요소들을 설명한다.

4.1.1. 변환 패턴

변환 패턴은 대상이 되는 소프트웨어 아키텍처가 목표가 되는 소프트웨어 아키텍처로 진화하기 위한 변환 과정을 정형화된 변환 명령어를 사용하여 정의한 것이다. 변환 패턴은 한번 정의가 되면 Java 언어의 method처럼 재사용될 수 있다. 이 경우, 대상이 되는 소프트웨어 아키텍처가 변환 패턴의 입력 값이 되고, 목표가 되는 소프트웨어 아키텍처가 변환 패턴의 출력 값이 된다.

```
transformation_ROS (Σ,Comps,OldDB,UpdatedDB)
    createComp(ReplicaDB);
    ReplicaDB=copyComp(OldDB);
    add(Σ,ReplicaDB);
    for each component c of Comps
        connect(Σ,ReplicaDB.Read, c.Read);
        disconnect(Σ,OldDB.Read, c.Read);
        disconnect(Σ,OldDB.Write, c.Write);
    end for
    createComp(UpdatedDB);
    delete(Σ,OldDB);
    add(Σ,UpdatedDB);
    for each component c of Comps
        connect(Σ,UpdatedDB.Read, c.Read);
        connect(Σ,UpdatedDB.Write, c. Write);
        disconnect(Σ,ReplicaDB.Read, c.Read);
    end for
    if UpdatedDB is successfully committed
        then delete(Σ,ReplicaDB);
        else rollback(Σ,Comps,OldDB,UpdatedDB);
```

[그림 3] 'Transformation_ROS' 변환 패턴 정의

예를 들어, [그림 3]은 Transformation_ROS라는 변환 패턴을 정의하고 있다. 이 변환 패턴은 Σ (대상이 되는 소프트웨어 아키텍처 전체를 의미), Comps(컴포넌트 셋을 의미), OldDB(진화 대상이 되는 DB 컴포넌트), 그리고 UpdatedDB(진화의 목표가 되는 DB 컴포넌트)를 입력 값으로 가지며, 출력 값은 OldDB 컴포넌트가 UpdatedDB 컴포넌트로 진화된 소프트웨어 아키텍처가 된다. 이 변환 패턴에 정의되어 있는 변환 과정은 Read-Only Service (이하 ROS)에 해당하는 변환 과정으로써, DB 컴포넌트를 진화시키는 동안에 DB에 접근하는 컴포넌트들이 read 명령어만 할 수 있도록 임시로 변경하여 진화 목표를 달성하는 변환 과정이다.

4.1.2. 소프트웨어 진화 과정

소프트웨어 진화 과정은 변환 패턴과 소프트웨어 아키텍처를 이용하여 대상 소프트웨어 시스템을 진화시키기 위한 과정을 의미한다. 소프트웨어 진화 과정은 [그림 4]와 같은 순서를 가진다.

아래의 진화 과정을 통해, 아키텍처 수준에서 논의된 소프트웨어 진화 과정이 실제 물리적 구현 수준에 적용이 되게 된다.

입력 값:
- 소프트웨어 아키텍처: 논리적 수준에서 소프트웨어 시스템을 ADL 등의 언어를 사용하여 표현함.
- 아키텍처-구현 매핑 정보: 논리적 수준의 구성요소들과 실제 구현요소들 사이의 관계를 표현함.
- 변환 패턴: 논리적 수준에서 변환 과정을 정의함.

출력 값:
- 진화 후의 소프트웨어 아키텍처: 논리적 수준에서 변환 패턴으로 인해 진화한 소프트웨어 아키텍처를 표현함.
- 구체적 변환 패턴: 진화를 적용하는 소프트웨어 시스템이 실제로 해석할 수 있도록 구체적인 구현 정보를 담고 있는 변환 패턴을 의미함.

프로세스:
1) 논리적 수준의 변환 패턴의 모든 입력 값들과 출력 값들을 실제 구현요소들로 변환해주어 구체적 변환 패턴을 만든다.
2) 구체적 변환 패턴을 해석하여 실제 소프트웨어 시스템을 진화시킨다.
3) 논리적 수준의 변환 패턴을 소프트웨어 아키텍처에 적용하여 진화 후의 소프트웨어 아키텍처를 만든다.

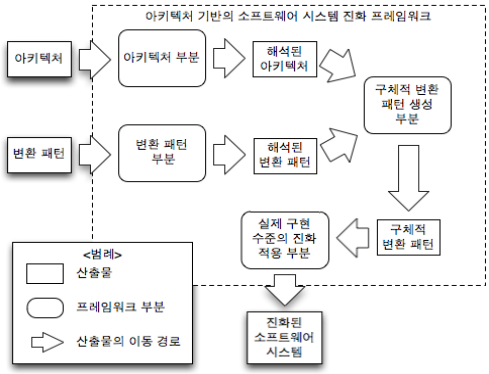
[그림 4] 소프트웨어 진화 과정

위의 진화 과정을 통해, 아키텍처 수준에서 논의된 소프트웨어 진화 과정이 실제 물리적 구현 수준에 적용이 되게 된다.

4.2 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크

3.1절에서 다룬 소프트웨어 진화 요소들을 이용한 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크는 크게 4부분으로 나뉠 수 있다.

- 아키텍처 부분: ADL, 혹은 그에 준하는 엄격하게 정의된 아키텍처 문서를 입력받아 내용을 해석(parse)하는 부분이다. 해석된 아키텍처를 출력하며, 해석된 아키텍처는 '구체적 변환 패턴 생성 부분'이 사용할 수 있도록 '아키텍처-구현 매핑 정보'를 반영한다.
- 변환 패턴 부분: 변환 패턴 표현 언어를 이용하여 정의된 변환 패턴을 입력받아 변환 패턴에 사용된 오퍼레이션들과 관련 입력 값들을 해석(parse)하는 부분이다. 해석된 변환 패턴을 출력한다.
- 구체적 변환 패턴 생성 부분: 아키텍처 부분에서 해석한 내용을 바탕으로 변환 패턴을 실제 구현 수준에서 적용될 수 있도록 구체적 변환 패턴으로 변경해주는 부분이다. 구체적 변환 패턴을 출력한다.
- 실제 구현 수준의 진화 적용 부분: 구체적 변환 패턴을 해석하여 실제 구현된 소프트웨어 시스템에 진화 과정을 적용하여 진화시키는 부분이다. [그림 5]는 이러한 프레임워크를 시각적으로 보여준다.



[그림 5] 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크

[그림 5]에서 볼 수 있듯이, 본 프레임워크에서는 소프트웨어 아키텍처와 변환 패턴을 입력값으로 받는다. 아키텍처 부분과 변환 패턴 부분은 서로 연관 관계가 없기 때문에 평행하게 실행 될 수 있으며, 둘의 결과를 '구체적 변환 패턴 생성 부분'에서 이용하여 구체적 변환 패턴을 생성한다. 구체적 변환 패턴을 읽고 해석하는 '실제 구현 수준의 진화 적용 부분'은 실제 소프트웨어가 배치되어있는 환경(프로그래밍 언어, 운영 체제 등)에 따라 달라진다. 즉, '구체적 변환 패턴'에 정의되어 있는 오퍼레이션들을 해석하여 실제로 적용하기 위해서는 같은 오퍼레이션이라도 사용 프로그래밍 언어, 운영 체제, 구현 모습 등에 따라 달라질 수 있다.

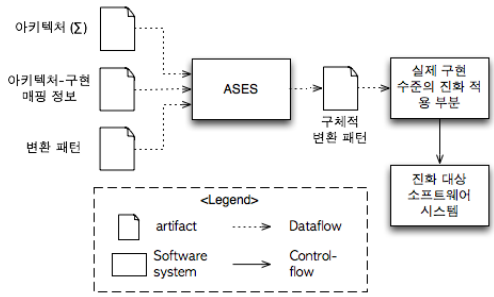
5. 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크 적용 예시

5.1절에서는 4.2절에서 설명된 프레임워크를 구현한 Automatic Service Evolution System (ASES)을 배경도(context diagram), 개념도(conceptual diagram), 컴포넌트 앤 커넥터 다이어그램(Component and Connector diagram)을 통해 소개한다. 그리고 마지막으로 5.2절에서 ASES가 어떻게 적용되었는지를 설명한다.

5.1 프로토타입 구현

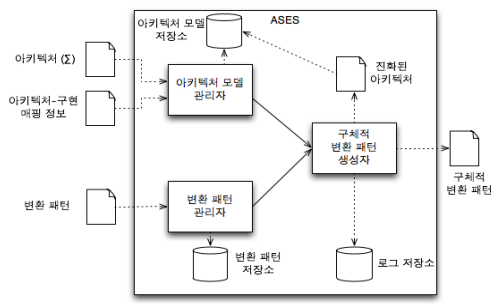
본 논문에서는 3.2절에서 설명한 아키텍처 기반의 소프트웨어 시스템 진화 프레임워크를 구현한 프로토타입을 제작하였다. 이 프로토타입은 Automatic Service Evolution System(이하 ASES)라고 부른다. ASES는 아키텍처 입력 값을 위해 카네기 멜론 대학교에서 개발된 Acme를 사용하였으며, AcmeStudio를 통해 작성된다. 또한, '아키텍처-구현 매핑 정보'를 독립적인 텍스트 기반의 파일에 저장을 하여 입력 값으로 받는다. 변환

패턴의 경우 동일한 방식으로 파일에 저장하여 입력 값으로 받는다. 출력 값은 '구체적 변환 패턴'으로 '실제 구현 수준의 진화 적용 부분'에서 해석할 수 있도록 자바 오브젝트 형태로 출력하게 된다. 마지막으로 '실제 구현 수준의 진화 적용 부분'은 자바 인터페이스 형태로 작성되어 각기 다른 구현 환경에 맞춰 개발 될 수 있도록 한다.



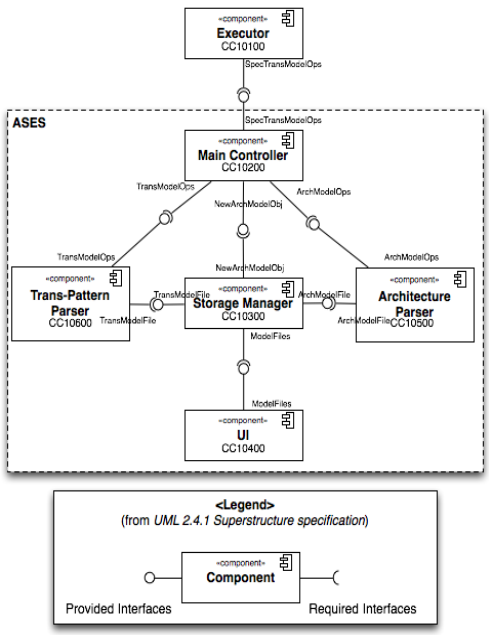
[그림 6] ASES의 배경도

[그림 6]은 ASES의 배경도를 나타낸다. 앞서 설명했듯이, 배경도에도 입력 값과 출력 값이 표현이 되어 있다. '실제 구현 수준의 진화 적용 부분'은 ASES 밖으로 빠져있으며 구현 환경에 따라 다른 모습으로 구현될 것이다. 배경도에서 볼 수 있듯이, ASES가 출력한 구체적 변환 패턴을 입력 받아 '실제 구현 수준의 진화 적용 부분'이 진화 대상 소프트웨어 시스템을 진화시킴을 알 수 있다.



[그림 7] ASES의 개념도

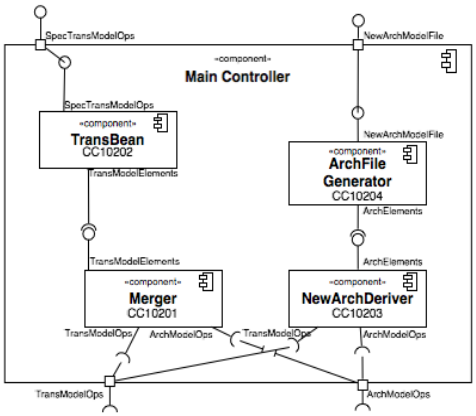
[그림 7]은 ASES의 개념도를 보여준다. 범례는 [그림 6]의 범례와 동일하다. 개념도에서 볼 수 있듯이, 아키텍처 모델 관리자는 입력 받은 아키텍처를 아키텍처 모델 저장소에 저장하고, 변환 패턴 관리자도 변환 패턴 저장소에 입력받은 변환 패턴을 저장한다. 이는 향후 재사용하거나 로그 기록을 남기기 위함이다. 구체적 변환 패턴 생성자는 해석된 아키텍처와 변환 패턴 내용을 바탕으로 구체적 변환 패턴을 생성하고, 관련 기록을 로그 저장소에 기록하며, 논리적 수준에서 진화된 아키텍처를 생성하여 아키텍처 저장소에 저장한다.



[그림 8] ASES의 컴포넌트 앤 커넥터 다이어그램

[그림 8]은 ASES의 구체적인 구현 이전에 설계한 컴포넌트 앤 커넥터 다이어그램이다. 각 컴포넌트들은 [그림 9]와 같이 상세하게 분해되어 기록되었다. [그림 7]의 개념도에서 나왔던 각종 저장소들은 Storage Manager를 통해 한 곳에서 관리되며, Architecture Parser와 Trans- Pattern Parser는 각각 아키텍처 모델 관리자, 변환 패턴 관리자의 서비스를 제공한다. Main Controller는 구체적 변환 패턴 생성자의 서비스를 제공하며, 기록을 남기고, 진화된 논리적 수준의 아키텍처를 생성

하여 Storage Manager에게 넘긴다. 출력된 구체적 변환 패턴은 Executor에게 전달되며, Executor는 이를 해석하여 실제 구현된 소프트웨어 시스템에 적용하는 역할을 한다.

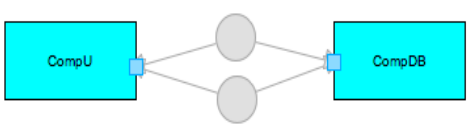


[그림 9] Main Controller (CC10200)이 분해된 모습

본 프로토타입에서는 Executor는 자바 인터페이스 형태로 작성되어 향후 서로 다른 환경에 맞춰 개발할 수 있도록 한다.

5.2 프로토타입 적용

본 프로토타입은 자바를 기반으로 작성되었으며, UI는 텍스트 기반의 콘솔 UI로 작성되었다. 본 프로토타입은 Acme로 작성된 간단한 아키텍처 파일과 '아키텍처-구현 매핑 정보' 파일, 그리고 변환 패턴 파일을 입력받아 구체적 변환 패턴을 텍스트 형태로 출력해주는 프로그램이다. 자바 인터페이스 형태로 구현된 Executor가 사용할 수 있도록 구체적 변환 패턴은 자바 오브젝트 형태로 저장된다.



[그림 10] 테스트에 사용된 Acme 기반의 아키텍처

[그림 10]의 아키텍처가 테스트에 사용이 되었다. 위의 아키텍처는 두 개의 컴포넌트와 2개의 커넥터로 구성되어 있다. 컴포넌트는 그림에서 보다시피, CompU와 CompDB이며, 커넥터는 각각 Read, Write이다.

변환 패턴은 [그림 3]의 transformation_ROS가 사용되었다.

```
*********ASES initial console-based UI**********
1. Work with a new system
2. Work with a recent system
0. exit
2
1.
2012.06.25
/Users/byron1st/Documents/workspace/nhn/data/System1.acme
/Users/byron1st/Documents/workspace/nhn/data/mapping_System1.dat
2.
2012.06.10
/Users/byron1st/Documents/ACME_workspace/Test_system/System1.acme
/Users/byron1st/Documents/ACME_workspace/Test_system/mapping_System1.dat
3.
2012.05.13
/Users/byron1st/Documents/ACME_workspace/Test_system/System_test.acme
/Users/byron1st/Documents/ACME_workspace/Test_system/mapping_System_test.dat
1
Type the path of a transformation pattern:
/Users/byron1st/Documents/workspace/nhn/data/testInput.dat
Is it correct? (/Users/byron1st/Documents/workspace/nhn/data/testInput.dat) (y/n)
y
```

[그림 11] ASES의 텍스트 기반 콘솔 UI

[그림 11]은 ASES의 텍스트 기반 콘솔 UI를 보여준다. Storage Manager를 통해 과거 사용했던 아키텍처, 변환 패턴을 불러올 수 있으며, 새로운 아키텍처, 변환 패턴으로 작업을 시작할 수 있다. [그림 11]은 [그림 2]에서 논리적 아키텍처 수준에서 표현되었던 transformation_ROS 변환 패턴이 실제 구현 수준에 적용되기 위해 구체적 변환 패턴으로 전환된 모습을 보여준다. 그림에서 볼 수 있듯이, 각 입력 값들은 모두 실제 구현 정보로 대체되었고, 구현 정보가 존재하지 않는 새로운 컴포넌트들은 <new>가 붙어 표현되었다. 이는 Executor에서 처리될 부분이다. 또한, for 구문으로 표현되어있던 부분은 아키텍처 모델을 해석한 내용을 바탕으로 판단한 횟수만큼 풀어서 표현되어 있다. 위 그림에서는 나타나있지 않지만, 'connect' 오퍼레이션은 모두 2번 반복된다. 왜냐하면, [그림 10]에서 CompU라고만 표현되어 있는 컴포넌트는 실제 구현상 2개의 컴포넌트로 구현되어 있으며, 이 점이 '아키텍처-구현 매핑 정보' 파일과 변환 패턴 파일에 담겨져 있기 때문이다. [그림 12]에서 나타난 모습은 일부이다.

```
transformation_ROS:
        /Users/byron1st/Documents/System1/
        <set>CompU
        /Users/byron1st/Documents/System1/CompDB/test1.java
        <new>UpdatedDB
<set>CompU:
        /Users/byron1st/Documents/System1/CompU/test1.java
        /Users/byron1st/Documents/System1/CompU/test2.java
createComp:
        <new>ReplicaDB
copyComp:
        /Users/byron1st/Documents/System1/CompDB/test1.java
        <new>ReplicaDB
add:
        /Users/byron1st/Documents/System1/
        <new>ReplicaDB
connect:
        /Users/byron1st/Documents/System1/
        /Users/byron1st/Documents/System1/CompU/test1.java&p&Read
        /Users/byron1st/Documents/System1/CompU/test1.java&p&Read
```

[그림 12] transformation_ROS의 구체적 변환 패턴 일부

6. 결 론

본 논문에서는 아키텍처 변화 패턴을 이용한 소프트웨어 시스템 진화 프레임워크를 제시하였다. 이를 위해, 아키텍처 기반의 진화 관련 연구인 Rainbow framework, Simplex architecture 등의 기존 연구들이 갖는 문제점들을 고찰하였다. 또한 아키텍처 변환 패턴 (Transformational Architecture Design)과 아키텍처 표현 언어인 ACME을 활용하여 아키텍처 기반의 소프트웨어 시스템 진화 프로세스를 제시하였다. 아키텍처 변화 패턴을 이용하여 직관적이고 도메인별로 다양하게 존재할 수 있는 소프트웨어 진화 패턴들을 지원할 수 있는 프레임워크를 제시함으로써 반복적인 소프트웨어 진화 작업의 자동화를 가능하게 하였다. 또한 ASES 라는 이름의 프로토타입을 설계하여 진화 과정을 실제로 적용한 예시로 구현하여 자동화가 가능한 진화의 가능성을 살펴보았다.

향후 연구로서, 소프트웨어 시스템 진화가 다양한 실제 구현 환경에 적용될 수 있도록 '실제 구현 수준의 진화 적용 부분'을 일반화하고 이를 실제 산업계에 적용할 것이다.

참 고 문 헌

[1] D. Garlan, S. Cheng, A. Huang, B. Schmerl P. Steenkiste. "Rainbow: Architecture-Based Self Adaptation with Reusable Infrastructure" In IEEE Computer, Vol. 37(10), October 2004.

[2] S. Cheng, D. Garlan and B. Schmerl. "RAIDE for Engineering Architecture-Based Self-Adaptive Systems," Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on, vol., no., pp.435,436, 16~24 May 2009

[3] I. Sommerville, "Software Engineering", 8th ed. Addison Wesley, 2006.

[4] P. Oreizy, N. Medvidovic, and R. N. Taylor, "Architecture-based runtime software evolution," in International Conference on Software engineering, ICSE '98. Washington, DC, USA: IEEE, 1998, pp.177~186.

[5] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self- adaptive software," IEEE Intelligent Systems, vol. 14, pp. 54~62, May 1999.

[6] M. M. Lehman, L. A. Belady, "Program evolution: processes of software change," Academic Press Professional, Inc., San Diego, CA, 1985

[7] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice. 2nd edition, Addison-Wesley Professional, 2012.

[8] E. M. Dashofy, A. v. d. Hoek, and R. N. Taylor, "A comprehensive approach for the development of modular software architecture description languages," ACM Trans. Softw. Eng. Methodol., vol. 14, pp.199~245, April 2005.

[9] L. Sha, R. Rajkumar, M. Gagliardi, "A Software Architecture for Dependable and Evolvable Industrial Computing Systems," CMU/SEI Technical Report, Jul. 1995.

[10] S. Kang, “Transformational Architecture Design,” KAIST Technical Report (CS-TR- 2011-359), Apr. 2012.

[11] R. van Ommering., R. van der Linden, F., Kramer, J. Magee, J., “The Koala Component Model for Consumer Electronics Software,” IEEE Computer 33(3), 2000.

[12] S. Cheng, D. Garlan, “Stitch: A language for architecture-based self-adaptation,” Journal of Systems and Software, Vol. 85, Issue 12, pp 2860~2875, December 2012.

[13] J. Dowling and V. Cahill, “The k-component architecture meta-model for self-adaptive software,” in REFLECTION ’01: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns. London, UK: Springer- Verlag, 2001, pp. 81~88.

[14] J. Zhang and B. H. C. Cheng, “Model- based development of dynamically adaptive software,” in ICSE ’06: Proceedings of the 28th international conference on Software engineering. New York, NY, USA: ACM, 2006,pp. 371~380.

[15] W. Cazzola, A. Ghoneim, and G. Saake, “Software evolution through dynamic adap- tation of its oo design,” in Objects, Agents and Features: Structuring Mechanisms for Contemporary Software, Lecture Notes in Computer Science. Springer-Verlag, 2004, pp. 69~84.

[16] D. Garlan, R. Monroe, and D. Wile, “Acme: An Architecture Description Interchange Language,” In Proceedings of CASCON’97, pp. 169~183, Nov. 1997.

[17] P. Kruchten. Architectural Blueprints-The “4+1” View Model of Software Architecture. IEEE Software 12, 42p~50p, 1995.

저자 소개

박 태 현

2010년 KAIST 전산학과 졸업(학사)
2012년 KAIST-CMU 소프트웨어공학 프로그램 졸업 (석사)
2012년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 공학, 소프트웨어 아키텍처, 소프트웨어 테스팅

안 휘

2010년 KAIST 전산학과 졸업(학사)
2012년 KAIST-CMU 소프트웨어공학 프로그램 졸업 (석사)
2012년~현재 KAIST 전산학과 박사과정
관심분야는 소프트웨어 아키텍처, 소프트웨어 제품라인 공학

황 상 철

2000년 경희대학교 산업공학과 졸업(석사)
2001년~2009년 삼성SDS 생산성혁신본부
2009년~2013년 NHN 기술혁신랩
2013년~현재 SK Planet Software Quality Engineering 팀

강 성 원

1982년 서울대학교 사회과학대학 졸업(학사)
1989년 University of Iowa 전산학 졸업(석사)
1992년 University of Iowa 전산학 졸업(박사)
1993년~2001년 한국통신(KT)선임연구원
1995년~1996년 미국 국립표준기술연구소(NIST) 객원 연구원
2001년~2005년 한국정보통신대학교 조교수
2003년~현재 Carnegie-Mellon University MSE프로그램 겸임교수
2005년~2009년 한국정보통신대학교 부교수
2009년~현재 KAIST 전산학과 부교수

박 종 빈

1994년 광운대학교 제어계측학과 졸업(학사)
2010년 University of Illinois at Urbana- Champaign 전산학 졸업(석사)
1995년~2008년 삼성 SDS
2008년~2012년 NHN 생산성혁신랩장
2012년~현재 SK Planet Software Quality Engineering 팀장

회원 가입 안내 및 회비 납부 요령

한국정보과학회 소프트웨어공학소사이어티는 회원 여러분에게 유익한 정보를 제공해 드리기 위하여 보다 충실한 내용의 논문지 발간 배포, 그리고 국제·국내 학술발표회 및 초청강연회와 단기강좌 등의 여러 가지 사업들을 추진하고 있습니다.

소프트웨어공학 소사이어티의 가입을 통해 정보 및 기술 교류, 그리고 인적 네트워크의 구성에 참여하시기를 기대합니다. 회원 가입을 위하여 아래의 회비 안내를 참고하시어 회비를 납부하시고, 다음 쪽의 입회원서를 작성하시어 아래 소프트웨어공학소사이어티 주소로 보내주시거나 팩스 또는 이메일을 통해 보내어 주시기 바랍니다.

한국정보과학회 소프트웨어공학소사이어티 연락처는 아래와 같습니다.

◈ **소프트웨어공학소사이어티**

주 소 : (우) 110-743 서울특별시 종로구 홍지문 2길 20, 상명대학교 소프트웨어대학관 G511 한국정보과학회 소프트웨어공학소사이어티 한혁수

전 화 : 02-2287-5033

팩 스 : 02-2287-0049

전자우편 : hshan@smu.ac.kr(한혁수 교수), jungwony@ajou.ac.kr(이정원 교수)

홈페이지 : http://www.sigse-kiss.or.kr/

◈ **회비 안내**

회원구분	•학생회원 : IT 분야 학과 또는 관심 있는 학생 •정회원, 종신회원 : IT 분야 종사자
가입비	학생회원, 정회원 : 20,000원, 종신회원 : 200,000원
년회비	학생회원, 정회원 : 20,000원

- 회비납부 방법

(1) 무통장입금 또는 계좌이체 후 입회원서 발송

: 계좌 번호 : 제일은행 150-20-358028 (이병정)

(2) 소사이어티 주관 학술행사 개최시, 행사장 당일 가입 및 납부 가능

개인회원용 입회원서

회원구분	학생회원 ( ) 정회원( ) 종신회원( )				
성명	한글		생년월일		
	영문				
연락처	직장전화		휴대전화		
	e-mail				
주소	직장명/부서		직급		
	직장주소	(우)			
학력	학사	년 월 - 년 월	대학교	과	
	석사	년 월 - 년 월	대학원	과	
	박사	년 월 - 년 월	대학원	과	
관심분야					

본인은 한국정보과학회 소프트웨어공학소사이어티의 취지에 찬성하여 회원으로 가입하고자 이에 입회원서를 제출합니다.

년 월 일

신 청 인: (인)

한국정보과학회 소프트웨어공학 소사이어티 회장 귀하

논문지 논문 모집 (Call for Papers)

한국정보과학회 소프트웨어공학 소사이어티에서는 매년 4회에 걸쳐 '소프트웨어공학 소사이어티 논문지'를 발간하고 있습니다. 이 논문지에는 소프트웨어공학 전반에 걸친 연구논문과 산업계 논문을 게재해 오고 있습니다. 다음과 같은 소프트웨어공학 주제에 관련된 논문을 모집하고 있으니 학계와 산업계의 여러분들의 적극적인 논문투고를 바랍니다.

◈ 논문 주제

○ 소프트웨어 설계 및 아키텍처
○ 소프트웨어 재사용 및 프로덕트라인
○ 요구공학
○ 소프트웨어 품질 및 테스트
○ 관리 및 프로세스
○ 소프트웨어 정형 기법
○ 서비스기반 소프트웨어 개발
○ 임베디드, 모바일, 웹기반 소프트웨어 개발
○ 기타 소프트웨어 응용 (국방, 자동차, 조선 등의 분야)

◈ 논문심사

○ 투고된 논문은 편집위원회에서 심사 선정하며, 필요 시 외부 심사위원을 위촉하여 심사를 합니다. 제출된 논문은 반환하지 않습니다.
○ 심사료 및 게재료: 없음

◈ 논문 제출

○ 소프트웨어공학 소사이어티의 논문지 투고 양식(http://www.sigse-kiss.or.kr/)을 사용하며, 논문의 분량은 10장으로 제한합니다.
○ 논문지 투고규정에 따라 작성된 심사용 논문파일은 온라인투고시스템을 통하여 투고하시기 바랍니다.

◈ 문의처 (편집위원회)

○ 편집이사 : 강성원 교수 (KAIST, 042-350-3512, sungwon.kang@kaist.ac.kr)
○ 편집이사 : 윤회진 교수 (협성대학교, 031-299-0841, hjyoon@uhs.ac.kr)
○ 편집이사 : 이관우 교수 (한성대학교, 02-760-5864, kwlee@hansung.ac.kr)
○ 편집이사 : 채흥석 교수 (부산대학교, 051-510-3517, hschae@pusan.ac.kr)
○ 편집이사 : 김문주 교수 (KAIST, 042-350-3543, moonzoo@cs.kaist.ac.kr)
○ 편집위원 : 김정아 교수 (관동대학교, 033-649-7801, clara@kwandong.ac.kr)
○ 편집위원 : 김현수 교수 (충남대학교, 042-821-6657, hskim401@cnu.ac.kr)
○ 편집위원 : 이우진 교수 (경북대학교, 053-950-6378, woojin@mail.knu.ac.kr)
○ 편집위원 : 박수진 교수 (서강대학교, psjdream@sogang.ac.kr)
○ 편집위원 : 이지현 교수 (대전대학교, jihyun30@dju.ac.kr)
○ 편집위원 : 최종무 교수 (단국대학교, choijm@dankook.ac.kr)
○ 편집위원 : 김태호 박사 (ETRI, taehokim@etri.re.kr)

투 고 요 령

1. 소프트웨어공학소사이어티 논문지에 실리는 원고는 주제 논문, 일반 논문, 산업체 기고 등으로 구분하며 다음과 같은 분야에 대하여 모집한다.
   가. 소프트웨어공학 및 그 응용분야에 대한 연구결과
   나. 강좌 및 관련 교육사항 소개 (목적, 과정, 일정, 대상, 특징)
   다. 소프트웨어 도구 및 방법론 소개 (가격, 특징, 종류, 적용사례)
   라. 소프트웨어 산업에 대한 학계, 업계의 주요 관심사
   마. 기타 관련 사항

2. 투고자는 원칙적으로 본 소사이어티의 회원으로 한다. 다만 공동 또는 초청 기고자는 예외로 한다.

3. 논문은 원칙적으로 한글로 작성한다.

4. 원고는 한글(hwp), 워드(MS Word), PDF 형식 중 하나를 택하여 A4용지에 작성하며, 그림과 표를 포함하여 10쪽 이내로 한다.

5. 논문 내용에 직접 관련이 있는 문헌에 대해서는 이들 문헌에 관련이 있는 본문 중에 참고문헌 번호를 쓰고 그 문헌을 참고문헌 난에 인용 순서대로 기술한다. 참고문헌은 학술지의 경우 저자, 제목, 학술지명, 권, 호, 쪽수, 발행 연도의 순서로, 단행본은 저자, 서명, 쪽수, 발행처, 발행 연도의 순서로 기술한다.
   [1]Cole, R., "Parallel Merge Sort", SIAM Journal of Computing, vol.17, No.4, pp.770-785, 1988.
   [2]김수형, 강명호, 조형제, 송주석. "안전하고 효율적인 침입자 역추적 시스템", 정보과학회논문지, 제25권, 제10호, pp.1123-1131, 1998

6. 논문은 소프트웨어공학 소사이어티(http://www.sigse-kiss.or.kr/)의 온라인 투고 시스템을 통해 제출한다.

7. 논문투고신청서를 반드시 작성하여 이메일(hjyoon@uhs.ac.kr)로 제출한다.

7. 원고 접수는 수시로 하며, 접수일은 온라인 접수일로 한다.

8. 기타 자세한 사항은 한국정보과학회 논문지 투고 요령을 따른다.

한국정보과학회 소프트웨어공학소사이어티 임원명단

구분		성 명	소속기관명	E-Mail
회 장		한 혁 수	상명대학교	hshan@smu.ac.kr
부회장 (기획)		권 기 현	경기대학교	khkwon@kyonggi.ac.kr
부회장 (편집)		강 성 원	KAIST	sungwon.kang@kaist.ac.kr
부회장 (학술)		홍 장 의	충북대학교	jehong@chungbuk.ac.kr
부회장 (조직)		이 병 걸	서울여자대학교	byongl@swu.ac.kr
부회장 (협력)		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
운영위원회	총무이사	이 정 원	아주대학교	jungwony@ajou.ac.kr
		유 준 범	건국대학교	jbyoo@konkuk.ac.kr
	기획이사	이 병 정	서울시립대학교	bjlee@uos.ac.kr
		서 주 영	아주대학교	jyseo@ajou.ac.kr
	조직이사	백 종 문	KAIST	jbaik@kaist.ac.kr
		김 영 철	홍익대학교	bob@hongik.ac.kr
	학술이사	인 호	고려대학교	hoh_in@korea.ac.kr
		고 인 영	KAIST	iko@kaist.ac.kr
	편집이사	윤 회 진	협성대학교	hjyoon@uhs.ac.kr
		이 관 우	한성대학교	kwlee@hansung.ac.kr
		채 흥 석	부산대학교	hschae@pusan.ac.kr
		김 문 주	KAIST	moonzoo@cs.kaist.ac.kr
	홍보이사	조 은 숙	서일대학교	escho@seoil.ac.kr
		이 찬 근	중앙대학교	cglee@cau.ac.kr
		박 용 범	단국대학교	ybpark@dankook.ac.kr
	협력이사	이 세 영	NIPA	sarahlee230@gmail.com
감사		차 성 덕	고려대학교	scha@korea.ac.kr
		이 상 은	NIPA	selee@nipa.kr
자문위원회		강 교 철	포항공과대학교	kck@postech.ac.kr
		권 용 래	KAIST	kwon@cs.kaist.ac.kr
		성 기 수	KISTI 고문	Kss13@truefriend.com
		배 두 환	KAIST	bae@se.kaist.ac.kr
		신 규 상	ETRI	gsshin@etri.re.kr
		양 승 민	숭실대학교	smyang@ssu.ac.kr
		우 치 수	서울대학교	wuchisu@selab.snu.ac.kr
		이 경 환	중앙대학교	kwlee@object.cau.ac.kr
		이 단 형	KAIST	danlee@cs.kaist.ac.kr
		정 기 원	숭실대학교	chong@comp.ssu.ac.kr
		박 수 용	서강대학교	sypark@sogang.ac.kr
		황 선 명	대전대학교	sunhwang@dju.kr
		전 진 옥	비트컴퓨터	jojeon@bit.co.kr
		김 수 동	숭실대학교	sdkim777@gmail.com
		이 긍 해	항공대학교	khlee@kau.ac.kr
		최 병 주	이화여자대학교	bjchoi@ewha.ac.kr

구분	성 명	소속기관명	E-Mail
이사	김 정 아	관동대학교	clara@kwandong.ac.kr
	남 영 광	연세대학교	yknam@yonsei.ac.kr
	박 수 진	서강대학교	psjdream@sogang.ac.kr
	염 근 혁	부산대학교	yeom@pusan.ac.kr
	오 재 원	카톨릭대학교	jwoh@catholic.ac.kr
	윤 희 병	국방대학원	hbyoon37@hanmail.net
	이 우 진	경북대학교	woojin@knu.ac.kr
	이 은 석	성균관대학교	leees@skku.edu
	이 석 원	아주대학교	leesw@ajou.ac.kr
	이 은 주	경북대학교	ejlee@knu.ac.kr
	전 태 웅	고려대학교	jeon@korea.ac.kr
	정 인 상	한성대학교	insang@hansung.ac.kr
	최 승 훈	덕성여자대학교	csh@duksung.ac.kr
	최 종 무	단국대학교	choijm@dankook.ac.kr
	최 호 진	KAIST	hojinc@kaist.ac.kr
	현 창 문	탐라대학교	cmhyun@tnu.ac.kr
	계 승 교	삼성SDS	seankae@samsung.com
	권 경 룡	국방기술품질원	ka-ja17@hanmail.net
	권 원 일	STA컨설팅	wonil@softwaretesting.co.kr
	김 상 기	현대자동차	sangkikim@hyundai-motor.com
	김 진 태	SEEG	jtkim@swexpertgroup.com
	민 경 오	LG전자	davidmin@lge.com
	민 상 윤	솔루션링크	sang@sol-link.com
	박 복 남	핸디피엠지	pbnkhb@hitel.net
	박 찬 규	국방SW산학연합회	milspark@hotmail.com
	배 현 섭	슈어소프트테크	hsbae@suresofttech.com
	손 세 창	인천공항공사	scsohn@airport.kr
	손 진 규	삼성탈레스	Jinkyu.son@samsung.com
	신 석 규	SW시험인증센터	skshin@tta.or.kr
	양 상 욱	KAI	sangyang@koreaaero.com
	유 영 수	현대엠앤소프트	ysyoo@hyundai-mnsoft.com
	윤 태 권	한국SW기술진흥협회	tkyune@empal.com
	윤 형 진	케피코	Hyoungjin.yoon@kefico.co.kr
	이 근	삼성전자	gskeun.lee@samsung.com
	이 성 남	방위사업청	dapalee@korea.kr
	이 우 복	삼성전자	woobok.yi@samsung.com
	이 장 수	한국원자력연구원	jslee@kaeri.re.kr
	장 주 수	모아소프트	jsjang@moasoftware.co.kr
	정 연 대	N3SOFT	ydchung@n3soft.co.kr
	조 병 인	국방과학연구소	chobyun@dreamwiz.com
	조 상 윤	다한테크	sycho@dahan.co.kr

2012~2013 소프트웨어공학소사이어티 논문지 편집위원회

소프트웨어공학소사이어티 논문지 제26권 제1호 (통권 97호)

발 행 일 ॥ 2013년 3월 31일
발 행 인 ॥ 한혁수
편 집 인 ॥ 강성원
발 행 처 ॥ 사단법인 한국정보과학회 소프트웨어공학소사이어티
연 락 처 ॥ 서울특별시 종로구 홍지문 2길 20, 상명대학교 소프트웨어 대학관 G511
한국정보과학회 소프트웨어공학소사이어티 한혁수
전 화 : 02-2287-5033, 팩 스 : 02-2287-0049
전자우편 : hshan@smu.ac.kr(한혁수 교수), jungwony@ajou.ac.kr(이정원 교수)
홈페이지 : http://www.sigse-kiss.or.kr/
인 쇄 처 ॥ (주)참기획 (전화 : 042-861-6380, 팩스 : 042-861-6381)