시나리오 기반 자기적응형 소프트웨어의 효율적인 분석 방안

백수진*, 송영재**
*경희대학교 컴퓨터공학과

e-mail:croso@khu.ac.kr

An Efficiency Analysis Method of Self-adaptive software based Scenario

Su-jin baek*, Young-jae Song**
*Dept of Computer Engineering, Kyung-Hee University

요 약

기존 컴퓨팅 패러다임에서는 고정된 환경을 가정하여 소프트웨어를 설계하였으므로 급속한 시장 환경의 변화와 소비자의 불확실한 요구조건에 대응하여 개발하기에 어려움이 있다. 따라서 외부 환경의 변화를 직면하였을 때 동작을 멈추는 것이 아니라 그 변화를 감지하고 대안을 선택하여 지속적으로 서비스를 제공할 수 있는 자기 적응형 소프트웨어가 필요하다. 그러나 기존의 자기적응형 소프트웨어에 대한 연구는 적응형 소프트웨어에 영향을 주는 문맥정보를 모델링하는 기법이나 적응을 위해 대체할 수 있는 기능들을 찾아내는 방법에 대한 연구가 부족한 실정이다. 이 문제를 해결하기 위해 본 논문은 시나리오를 이용한 목표 기반으로 분석을 하고, 분석된 요구사항의 가변 수와 크기에 따라 프로그램의 문법뿐 아니라 사용자의 관점에서도 의미 있도록 프로그램 동적 슬라이싱 기법을 적용하도록 한다. 또한, 제안된 방법이 전 과정에 문맥에 대한 분석, 설계 정보가 반영되어 동적으로 재구성하는 방법을 제시하도록 한다.

1. 서론

기존 컴퓨팅 패러다임에서 개발자들은 고정된 환경을 가정하여 요구사항 분석하고, 실행 환경을 잘 정의하여 소프트웨어를 설계하였다. 그러나, 급속한 시장 환경의 변화와 소비자의 불확실한 요구 조건이 많아짐에 따라 이를 대응하기 위해 모든 상황들을 완벽하게 분석하고 설계하는 것은 어려운 일이다. 따라서 소프트웨어는 외부 환경의 변화를 직면하였을 때 동작을 멈추는 것이 아니라 그 변화를 감지하고 대안을 선택하여 지속적으로 서비스를 제공할 수 있는 자기 적응형 소프트웨어가 될 필요가 있다.

자기 적응형 소프트웨어는 자신의 행위를 평가하고 소프트웨어의 의도대로 수행하고 있지 않다고 판단되거나 더 나은 성능으로 향상시킬 수 있다고 판단될 때, 자기 적응형 소프트웨어는 자신의 행위를 변경한다[1]. 자기 적응형 소프트웨어를 구현하기 위해서는 실행 중인 소프트웨어의 상태를 모니터링과 소프트웨어에게 발생된 문제를 분석하고 적응의 필요성을 판단하는 평가, 발생된 문제를 해결하기 위한 적응 전략을 찾아내고 이에 따라 실행중인 소프트웨어의 구조와 행위를 동적으로 재구성을 해야 한다[2]. 그러나 현재 적응형 소프트웨어에 대한 연구는 소프트웨어가 외부 환경의 변화에 '어떻게' 적응하도록 할 것인가에 대한 적응 메커니즘에 초점이 맞추어져 있으며, 적응형 소프트웨어에 영향을 주는 문맥정보를 모델링하는 기법이나 적응을 위해 대체할 수 있는 기능들을 찾아내는 방법에 대한 연구가 부족한 실정이다. 이를 위해서는 기존 시스템에서의 문맥정보와 외부 상황에서의 새로운 요구사항에 대한 문제 파악을 하여 변경하고자 하는 행위를 여러 관점에서 판별하도록 해야 한다. 기존의 소프트웨어 분석 설계 방법들은 각기 장, 단점을 가지고 있기 때문에 하나의 분석 기술만을 사용하여 요구 사항을 분석한다면, 완벽한 요구 사항 분석을 이루지 못해 사용자의 정확한 요구에 맞는 소프트웨어 개발이 어렵다. 또한, 이들은 각 단계에서 수행하는 세부 활동과 산출물을 제시하고 있으나 문맥의 변화가 적절히 반영되지 못하고 있다.

본 논문은 시나리오를 이용한 목표 기반으로 분석을 하고, 분석된 요구사항의 가변 수와 크기에 따라 프로그램의 문법뿐 아니라 사용자의 관점에서도 의미 있도록 프로그램 슬라이싱 기법을 적용하도록 한다. 또한, 제안된 방법이 전 과정에 문맥에 대한 분석, 설계 정보가 반영되어 동적으로 재구성하는 방법을 제시한다.

본 논문은 다음과 같이 구성되었다. 2장에서는 관련연구에 대해서 기술하고, 3장에서는 제안한 시스템의 전반적인 프레임워크를 기술하고, 마지막으로 4장에서는 결론과 향후 연구 과제에 대해서 기술하였다.

2. 관련연구

2.1 시나리오를 이용한 목표 기반 분석

시나리오를 이용한 목표 기반 분석은 목표를 성취하기 위한 시나리오 생성 방법을 제공하고 목표들 간의 관계 설정을 위한 방법을 제공한다[3]. 시나리오 생성을 통해 목표를 발견하고, 발견된 목표는 다시 시나리오를 생성시킨다. 그러나 시나리오를 이용한 목표 기반 분석은 초기 목표 식별이 어렵고, 어느 단계에서 목표 정제가 중단되어야 하는지에 대한 기준이 존재하지 않기 때문에 요구사항 추출, 분석이 효율적이지 못할 수도 있다.

2.2 자기적응형 소프트웨어에 대한 기존 연구

자기적응형 소프트웨어란 자신의 행위를 평가하고 소프트웨어의 의도대로 수행하지 못하거나, 더 나은 성능으로 향상시킬 수 있다고 판단될 때, 자기 적응형 소프트웨어는 자신의 행위를 변경한다. 자기적응형 소프트웨어에 대한 연구로써 MIT의 DDA(Dynamic Domain Architecture)와 CMU(Carnegie-Mellon University)의 Rainbow Project 등이 있다. MIT의 DDA(Dynamic Domain Architecture)는 도메인 내의 모든 어플리케이션이 가지고 있는 공통적인 기능과 가변성을 표현하는 도메인 아키텍처 개념을 자기적응형 소프트웨어에 적용하고 있다. 그러나 자기적응형 소프트웨어의 적응에 있어서 코드와 알고리즘 대체를 통한 접근 방식을 취하고 있다[5]. Rainbow Project는 소프트웨어 시스템의 적응을 위해 소프트웨어 아키텍처와 재사용 가능한 기반을 사용하는 Rainbow framework을 제시한다[6]. 이러한 자기 적응형 소프트웨어에 대한 연구들은 소프트웨어가 외부 환경의 변화에 '어떻게' 적응하도록 할 것인가에 대한 적응 메커니즘에 초점을 두고 있다. 다시 말하면, 적응형 소프트웨어에 영향을 주는 문맥정보를 모델링하는 기법이나 적응을 위해 대체할 수 있는 기능들을 찾아내는 방법에 대한 연구가 부족하다.

2.3 가변성 관리 방법

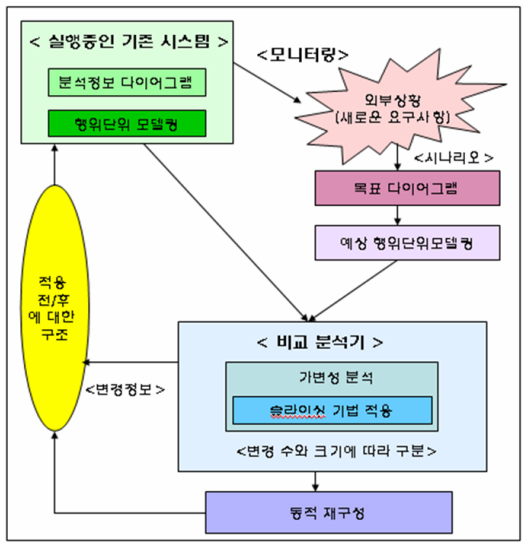
가변성 관리 방법에는 Parameterization, Inheritance, Information Hiding, Variation Point 가 있다. 이들 중 Parameterization, Information Hiding 방법은 외부상황의 새로운 요구사항에 맞도록 동적으로 재구성하는데 적합하다. Inheritance 방법은 결정에 따라 가변적으로 제공될 대상이 동일한 인터페이스를 따르지 않는 경우에 적용될 수 있는 방법이다. 또한 이것은 오퍼레이션의 추가 및 변경 등을 통해 가변성이 제공될 수 있고, KobrA, PuLSE에서 사용하나 동적으로 재구성에서는 적합하지 않다. Variation Point 방법은 Variation Point 들의 집합으로 구성된 핵심 자산 컴포넌트를 이용한다. 개발자는 가변점으로부터 파생된 고유의 가변들을 선택함으로써 대상 시스템 컴포넌트를 구축할 수 있다. 개발자에게 고유의 가변들을 생성하고 유지 관리하는데 있어 높은 유연성을 제공한다. 다른 방법들과의 차이점은 가능한 모든 가변들을 제공하고 개발자가 사용하고자 하는 방향대로 선택할 수 있도록 하나 예상된 가변들을 제외한 추가적인 경우 발생 시 유연하게 대응하기 힘들다.

3. 슬라이싱 기법을 응용한 자기적응형 소프트웨어의 효율적인 분석 방안

본 장에서는 먼저 제안하는 프레임워크의 전체적인 구성에 대해서 설명하고, 프레임워크의 관련자별 역할과 외부상황 분석 방법, 제어 및 모니터링 소프트웨어를 통한 동적 재구성 과정에 대해서 설명하도록 한다.

3.1 프레임워크 구성

본 논문에서 제안하는 전체적인 프레임워크는 아래 <그림 1>과 같이 실행중인 기존 시스템에 대해 시나리오를 기반으로 하여 분석 정보 다이어그램과 행위단위를 모델링한다. 또한 모니터링을 통해 외부상황에서의 새로운 요구사항을 동적으로 적용시키기 위해 시나리오를 기반으로 하여 목표 그래프를 작성하고 필요한 행위를 예상하여 모델링한다. 저장된 실행중인 기존 시스템에서의 분석 정보 다이어그램과 행위단위 모델링 정보와 외부상황에서의 목표 그래프와 예상 행위 모델링을 통해 비교분석기를 통해 분석을 한다.



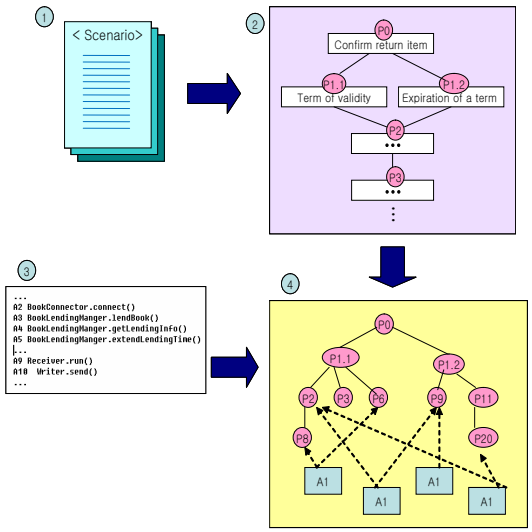
<그림 1> 전체적인 프레임워크

비교 분석기는 모니터링을 통한 내부, 외부 내용을 이용하여 가변성을 분석한다. 가변성 분석에서 초점을 둔 내용은 문맥상의 변경수와 크기에 따라서 적절한 대응을 한다.

값만 변경 요구 될 때는 구조 변경이 필요 없지만 컴포넌트 교체 또는 전체적으로 기존 구조 변경이 필요할 시에는 우선 동적 슬라이싱 기법을 적용하여 정리를 한 후 적합한 방법으로 재구성을 한다. 또한 각각의 변경 사항이 있을 시 적용 전과 후에 대한 정보를 주기적으로 저장하여 전 과정에 정보가 반영되도록 한다.

3.2 실행중인 기존 시스템에서의 모니터링 정보

실행중인 기존 시스템에서의 분석 정보 다이어그램은 <그림 2>과 같이 시나리오를 기반으로 정적 분석을 하여 정보를 추상화한다. 또한 이때 기존 시스템의 우선 순위 정보도 고려하여 함께 반영하도록 한다. <그림 3>은 도서관리 시스템에서의 반납 시나리오에 대한 예이다. 시나리오를 통해 주요 기능을 수행하는 핵심 메서드들로만 내용을 추출하여 분석 정보 다이어그램으로 작성한다. 이 내용을 토대로 행위 단위를 분석하고, 분석 정보 다이어그램과 행위단위들 간의 관계를 파악하여 모델링한다.



<그림 2> 기존 시스템의 시나리오 기반 정보 추출

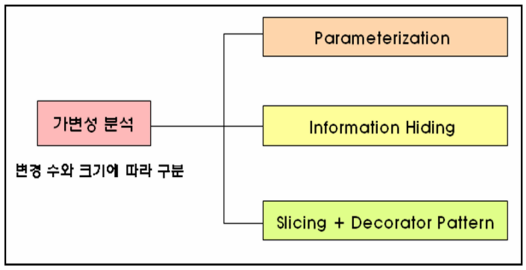
이 때, 행위단위 모델링에서의 행위단위는 클래스의 메서드로 정의한다. 클래스 다이어그램과 같이 클래스와 메서드를 모델링하고, 행위단위들 간에 선후관계를 파악하여 구조화하도록 한다. 또한, 효과적인 성능을 위해서 중요한 행위 단위들만 모델링 하도록 한다.

3.3 외부상황의 모니터링 정보

외부 모니터링을 통해 새로운 요구사항이 발생하면 시나리오를 기반으로 추상화하여 목표 다이어그램을 그리도록 한다. 이때 직접적으로 자연어로 작성된 새로운 요구 사항 문제 기술문과 기존의 객체 지향 요구 명세를 비교 검증하는 것은 불합리하기 때문에 문제 기술문으로부터 내용을 추출하고 정보 트리형태로 예상 행위 단위를 모델링한다. 실행중인 기존 시스템에서의 정보를 추출하여 분석 정보 다이어그램과 행위단위 모델리오 같은 절차로 목표 다이어그램과 예상 행위 단위를 모델링한다. 이때 필요한 예상 행위 단위는 문장의 실행 구조를 파악하기 위해서 제어문을 포함하고 있는 문장의 결합 구조를 고려하여 반영시켜야 한다. 유사한 부류의 목표들은 상위로 올라 갈수록 통합된 목표를 지니게 되고 하위 목표와 시나리오의 의미를 포괄하는 언어로 기술하도록 한다. 시나리오와 목표 다이어그램을 통해 예상 선/후 행위단위에 초점을 맞춰 추상화하여 모델링한다.

3.4 비교분석기

실행중인 기존 시스템에서의 산출물인 분석정보 다이어그램과 행위단위 모델링 정보와 모니터링을 통해 얻어진 외부상황 목표 다이어그램, 예상 행위단위 모델링을 비교분석한다. 위의 모델링을 통해 새로운 요구사항의 범위, 문제들 간의 관계 등을 파악할 수 있다. 비교 분석기에서는 기존 시스템의 정적 분석을 한 다이어그램과 목표 다이어그램을 추상 문맥 트리 구조의 각 노드를 디자인 패턴을 이용한 트리 횡단 루틴을 이용하여 방문하는 방법으로 비교분석을 수행할 수 있도록 한다. 이때, 유사성을 최대한으로 찾아내어 변경하고자 하는 수를 줄이도록 해야 한다. 유사도 알고리즘은 기존 소스 코드에서의 유사성 분석이 아닌 추상화 개념의 구문트리 구조를 이용하여 유사도를 평가한다. 이때 유사성 분석 결과를 통해 변경하고자 하는 수와 크기에 따라서 적절하게 재구성되어야 한다.



< 그림 3 > 변경 수와 크기에 따른 가변성 분석

가변성의 변경 수와 크기에 따라 <그림 3>와 같이 크게 세 가지 기법으로 나누어 관리한다.

첫 번째 방법은 분석된 내용이 핵심 자산의 속성 값만 변경을 요구할 때의 경우로 Parameterization 의 가변성을 결정하는 메커니즘을 이용한다. 인수화된 속성 값을 변경시키거나 초기화시키기 위한 컴포넌트의 인터페이스가 존

재함으로써 가능하다.

두 번째 방법은 컴포넌트 자체를 다른 컴포넌트로 바꾸거나 추가할 경우로 Information Hiding의 가변 관리를 이용한다. 동일한 인터페이스를 가진 여러 버전의 컴포넌트를 이용하여 처리하는 방법으로 변화되는 범위가 각 컴포넌트 단위 또는 그 이내로 제한되는 경우나 인터페이스가 변하지 않아야 하는 경우에 적용될 수 있다.

세 번째 방법은 구조 변경이 요구될 시에 적용될 경우로 program slicing 기법을 통해 필요 없는 부분을 정리한 후 Decoration Pattern 기법을 이용하여 재구성을 한다.

기존 시스템은 유지보수를 거치는 동안 프로그램의 크기가 증가하며, 구조가 불건전해지고, 이해하기도 어려워졌다. 따라서 이 상태로 새로운 요구사항이 더해진다면 이전의 문제를 잠시 해결한 듯 해보이나 결과적으로는 나중에는 더 큰 문제를 가져올 수 있다. 따라서 시스템의 구조를 응집도를 최대화 하고 모듈간의 결합도를 최소화함으로써 향상될 수 있도록 문법에 기반할 뿐 아니라 사용자의 관점에서도 의미있는 방식으로 변경하도록 한다. 이때 선행-후행 조건을 이용하여 꼭 필요한 내용만을 재사용 가능하도록 컴포넌트로 추출하고 추가할 요구사항이 있다면 Decoration Pattern 기법을 이용하여 재구성 될 수 있도록 한다.

이 모든 변경 내용은 변경 전과 후로 각각 다르게 저장되도록 해야 한다. 변경한 후 이전 상태의 정보가 필요하거나 잘못 적용이 되었을 시에는 적용 전과 후 구조를 참고한다.

4. 결론 및 향후과제

본 연구에서는 실행중인 기존 시스템에서의 외부 상황에서 새로운 요구사항이 발생했을 시 적절하게 동적으로 재구성이 되도록 기존 시스템에서의 시나리오 분석을 통해 분석 정보 다이어그램과 행위단위 모델링을 추출하고 외부 상황에서의 목표 다이어그램과 예상 행위 단위 모델링을 추상화하여 비교 분석하였다. 유사성 정보에 따라 요구사항의 가변 수와 크기에 따라 프로그램의 문법뿐 아니라 사용자의 관점에서도 의미 있도록 프로그램 슬라이싱 기법을 적용하도록 한다. 또한, 제안된 방법이 전 과정에 문맥에 대한 분석, 설계 정보가 반영되어 동적으로 재구성하는 방법을 제안하였다.

그러나, 제안한 방법은 시나리오를 기반으로 하고 있기 때문에 실행 중인 기존 시스템과 외부 상황에서의 요구사항에 대해서 문서화되어 있어야 한다. 또한 유사성 분석을 위해 작성된 목표 다이어그램 작성 시 초기 목표 식별이 어렵고, 추출된 요구 사항을 검증할 수 있는 방안이 미흡하다. 비교 분석에서의 가변성의 크기와 수에 따른 관리를 위해 크기와 수에 대한 기준이 정의가 필요하다.

앞으로의 연구 방향은 부분적인 시나리오 기반 분석 뿐 아니라 전체적인 여러 시나리오를 기반으로 한 관계 분석과 유사도 평가, 변경에 따른 영향력 추출에 대해서 연구할 계획이다. 또한 변경 요구 사항에 대한 적정 범위 산출 방법에 대해서도 고려할 것이다. 마지막으로, 시나리오를 기반하여 목표 다이어그램과 행위단위에 대한 체계적인 작성, 요구사항의 관계 및 검증을 통합적으로 지원하도록 해야한다. 이를 위해서는 외부 상황의 변화에 적절하게 대응하여 동적으로 재구성 될 수 있는 자동화 도구 개발이 필요하다.

참고문헌

[1] “Self adaptive software", December, 1997. DARPA, BAA 98-12, Proposer Information Pamphlet, www.darpa.mil/ito/Solicitations/PIP_9812.html

[2] David Garlan, and Bradley Schmerl, "Model-based Adaptation for Self-Healing Systems," ACM SIGSOFT Workshop on Self-Healing Systems(WOSS'02), November 18-19, 2002.

[3]C. Rolland, C. Souveyet, C. Ben Achour, "Guiding Goal Modelling using Scenarios", IEEE Transactions on Software Engineering, Special Issue on Scenario Management, Vol. 24, No. 12, Dec. 1998. pp 1055-1071.

[4]Anind K.Dey, Daniel Salber and Gragory D. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications". Anchor article of a special issue on context-aware computing in the Human-Computer Interaction(HCI) Journal, Vol. 16 No. 2-4, pp.97-166, 2001.

[5]Robert Laddaga, Paul Robertson, Howard E. Shrobe, "Probabilistic Dispatch, Dynamic Domain Architecture, and Self-Adaptive Software", Self-Adaptive Software, Second International Workshop, IWSAS 2001, LNCS 2614, Springer-Verlag, pp.227-237, 2003.

[6] Shang-Wen Cheng, An-Cheng Huang, David Garlan, Bradley Schmerl, and Peter Steenkiste, "Rainbow: Architecture-Bases Self Adaptation with Reusable Infrastructure", IEEE Computer VOL. 37 No. 10, 2004.