다중 모델 기반의 자가 적응형 시스템

이상희[1],정철호[2],이은석[3]
성균관 대학교 컴퓨터공학과[1 2 3]
{neomine[1],jesus98[2],eslee[3]}@selab.skku.ac.kr

Multi-Aspect Model based Self-Adaptive System

Sanghee Lee[1], Chulho Jung[2] , Eunseok Lee [3]
Dept. of Computer Engineering, Sungkyunkwan University[1 2 3]

요 약

본 논문에서는 구조, 행위, 리소스, 환경의 여러 관점을 적용한 다양한 모델들을 이용하는 적응 프레임워크를 제안한다. 또한, 대상 시스템에 대해 앞에서 언급한 4 가지 모델을 위한 모델링 방법론과 각 모델링 요소들에 대한 효과적인 표기법을 제시하였다. 다양한 모델들을 통해 시스템의 구성 요소들 간의 관계 구조와 시스템의 계층적 상태와 행위 정보, 실행 환경을 구성하는 시스템 의존적인 요소 및 독립적인 요소까지의 정보들이 표현된다. 이들 모델간의 유기적인 상호 운용으로 통합적인 추론과 보다 정확한 평가가 가능하다. 이를 통해 시스템은 예상치 못한 변화에 대해 통합된 관점의 더욱 정확한 진단과 반영할 수 있다. 이를 기반으로 다양한 수준에서 적응 동작의 조절을 수행함으로써 하이브리드하고 보다 확장된 적응이 가능해진다. 논문에서 정의한 모델과 제안 프레임워크는 다른 도메인으로 재사용이 가능하다. 제안 시스템은 평가를 위해 프로토타입을 구현하여 원격 화상 회의 시스템에 적용하였으며, 그 기능과 유효성을 확인하였다.

Keyword : Self-adaptive, Multi-aspect Model, Modeling, Ubiquitous computing

1. 서론

최근 컴퓨팅 및 네트워크 기술의 급속한 발전으로 디바이스, 플랫폼, 네트워크 환경 및 사용자에 대한 요구 사항이 점차 다양해지고 있다. 이러한 다양함에 따라 각 객체들의 행동이 복잡해지며, 그들 사이의 관계 및 상호 작용도 복잡해지고 있다. 따라서, 소프트웨어는 예상하지 못한 외부 환경 변화에 직면하였을 때, 지속적으로 서비스를 유지하며 동시에 동적인 변경에 대한 적응 기술의 필요성이 증대되고 있다.[1] 이와 함께 시스템 스스로가 자신에게 영향을 미치는 외부 환경에 대한 정확한 이해와 정의를 위하여, 시스템의 여러 가지 환경 요소를 미리 모델링 하는 기술 또한 필요하게 되었다. 모델을 기반으로 한 기존의 대표적인 연구로는 아키텍쳐 모델 기반의 적응형 시스템이 있다.[2,3] 그러나, 아키텍쳐와 같이 한 가지 관점의 모델만을 기반으로 하는 기존 연구는 아키텍쳐 수준의 모델링 기법이기 때문에 시스템에서 중요한 모든 측면들을 종합적이고 엄밀하게 기술하기에는 부족하다는 단점이 있다. 또한, 시스템의 행위와 시스템 외부적인 요소까지 포함하는 통합된 진단이 어려우며, 시스템을 구성하는 컴포넌트들의 동적 조합만을 수행하여 적응하는 한계성을 가지고 있다. 객체의 행위를 모델링 하여 객체들이 객체 상호 간의, 또는 사용자와의 상호작용하는 활발한 동적 특성을 반영과 시스템의 동작에 영향을 줄 수 있는 모든 내부적, 외부적 요소에 대한 파악이 요구되며, 동적인 측면에서의 적응성도 필요하다. [4]

따라서, 본 논문에서는 기존 연구의 한계성을 보완하기 위해 시스템의 아키텍쳐에 관한 모델링과, 시스템의 행위적 요소에 대한 모델링, 시스템

자원에 대한 모델링, 그리고 시스템의 환경에 대한 모델링을 함께 수행하여, 이를 기반으로 다양한 시스템 변경 요인에 적응하는 프레임워크를 제안한다. 또한, 대상 시스템에 대해 앞에서 언급한 네 가지 모델에 대해 설명하고, 이들에 대한 모델링 방법론과 각 모델링 요소들에 대한 효과적인 표기법을 제시한다. 제시한 다양한 모델들을 통해 시스템의 구성 요소들 간의 관계 구조와 시스템의 계층적 상태와 행위 정보, 실행 환경을 구성하는 시스템 의존적인 요소 및 독립적인 요소까지의 정보들을 표현이 가능하며, 이들 모델간의 유기적인 상호 운용을 통해 통합적인 추론과 보다 정확한 평가가 가능하다. 시스템은 예상치 못한 변화에 대해 통합된 관점의 더욱 정확한 진단과 반영할 수 있다. 이를 기반으로 다양한 수준에서 적응 동작의 조절을 수행함으로써 보다 확장된 적응이 가능해진다. 논문에서 정의한 모델과 제안 프레임워크는 다른 도메인으로 재사용이 가능하다.

본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구를 살펴보고, 3 장에서는 모델을 기반으로 한 시스템에서 고려해야 할 컨텍스트의 정의와 모델링 방법을 제안한다. 또한, 모델 기반의 적응화를 위한 모듈을 제안하며, 5 장에서는 프로토타입을 구현과 시스템 평가를 하였으며, 6 장에서는 결론을 기술하였다..

2. 관련 연구

Self-Adaptive 소프트웨어는[5] 프로그램 스스로 변화를 이해하고, 모니터하고 수정 가능하게 하는 기술을 가진 소프트웨어로써 프로그램이 스스로 무엇을 해야 하고, 어떻게 그것을 하고, 어떻게 그 자신의 수행 성능을 평가하고, 그래서 어떻게 상태 변화에 응답하는지에 대한 내용을 이해하고 있어야 한다. 따라서, 소프트웨어가 환경에 적응하기 위해 다양한 모델을 이용하여 적응하는 연구들이 제안 되고 있으며, 그 중 대표적인 방법으로 아키텍쳐를 기반으로 동작하는 시스템을 들 수 있다. 아키텍쳐 기반의 자가 적응형 소프트웨어는[1,6] 사용자의 실행 환경과 변화에 맞추어 동적인 재배치와 재설정이 가능하게 하기 위해 아키텍쳐를 이용하여 변경하는 소프트웨어 이며, 실행 시에 스스로 적응하게 하기 위한 자가 적응형 소프트웨어에 대한 접근 방법 중에 하나이다. 상호 작용하는 컴포넌트들을 그래프 형태로 표현하는 소프트웨어 아키텍쳐를 변경함으로써 환경 적응은 효율적지만 사용자의 모든 주변 자원을 인식하고, 반영하지 못하는 단점이 있다.

3. 제안 시스템

본 연구에서는 실제 환경과 그 환경에서 동작되는 시스템의 적절한 모델들의 유기적인 상호 운용을 통하여, 모델과 실제 환경의 차이를 발견하면 모델을 바탕으로 문제의 본질을 진단하고 실행 중인 시스템의 재구성을 통하여 더욱 강건하게 작동되도록 하는 것이 본 시스템의 목표이다.

다양한 모델을 도입함으로써, 기존의 시스템들보다 보다 여러 관점을 통해 시스템이 인식 할 수 있으며, 유기적인 상호 운용을 통해 통합적인 진단과 적응 전략을 수행함으로써, 다양한 수준에서 적응 동작의 조절이 가능한 보다 확장된 적응을 수행할 수 있다. 3-1 절에서는 제시한 모델들에 대한 설명과 전체적인 프로세스 과정에 대한 설명을 기술한다. 3-2 절에서는 모델을 기반으로 동작하는 Change management Framework 의 구성에 대해 설명한다.

3-1. 다양한 관점의 모델링(Multi Aspect-Modeling)

모델링은 복잡한 시스템과 외부 환경을 이해하고, 체계적으로 분석하는데 있어 유용하게 사용되고 있다. 본 논문에서는 시스템의 전체적이고 다양한 관점을 이용하여 모델링을 수행하였으며, 이를 통해 시스템의 정적, 동적 특성과 외부의 환경 요소 등을 확장하여 표현하였다. 각 모델은 시스템이 개발되기 이전 단계에서 도메인에 대한 이해와 사용자의 요구사항에 대한 분석을 통해 기술 요소들이 식별되고 명세화된다. 명세화된 모델을 기반으로 실행 중인 시스템은 컨텍스트들을 수집하고,

분석을 통해 시스템과 외부 환경에 대한 변경이나 변화를 발견한다. 제시하는 모델은 아키텍쳐 모델, 행위 모델, 시스템 리소스 모델, 외부 환경 모델의 네 가지로 구성된다. 각 모델에서 기술하는 요소들과 내용은 다음과 같다.

1. Architecture Model

컴포넌트들이 서로 정확하게 결합하여 작동할 수 있는 기반 구조를 기술하는 정적 모델이다. 소프트웨어의 행위들을 실제로 수행되기 위해서 필요한 시스템의 구성요소인 컴포넌트들과 그들간의 물리적인 관계를 표현한다. 시스템의 초기 단계에 구성되어야 하는 컴포넌트 정보와 커넥터 정보, 커넥터의 인터페이스 정보, 컴포넌트간의 의존성과 같은 정보가 기술된다. 이를 기반으로 시스템의 실행 시에는 구동 중 에 추가된 새로운 컴포넌트 정보와 제거된 컴포넌트 정보, 컴포넌트 간의 새로운 연결 정보들을 수집하여 정의된 모델과 비교 분석을 수행함으로써 변화에 대한 진단을 수행한다. 이러한 요소들을 기술하기 위해 아키텍쳐 기술 언어(Architecture Description Languege)를 사용하여 모델링하였다.

2. Behavior Model

목표로 하는 서비스를 제공하기 위해 시스템이 시간 흐름적인 변화나 순서에 따라 내부적으로 일어나거나 유지해야 하는 모든 활동들을 기술하는 동적 모델이다. 이를 위해 시스템의 동적인 측면을 표현하는데 일반적으로 사용되는 활동 다이어그램을 확장하여 기술하였다.

시간 순서적 변화나 흐름에 따른 시스템 전체적인 관점의 상위 수준의 행위를 기술하며, 이들은 서브 행위들을 표현할 수 있는 계층 구조로 기술된다. 따라서, 시스템 전체적인 관점에서의 행위의 표현이 가능하며, 이와 동시에 시스템을 구성하는 모든 개체들 각각이 내부적으로 갖는 시간적 흐름에 따라 변화되는 모든 활동들이 기술된다.

추가적으로, 전체적인 관점에서부터 상세한 활동까지 기술한 모든 행위 모델들은 각각의 활동에 대해서 성능 평가 요소들이 함께 기술된다. 이를 통해, 다양한 적응 전략에 한 성능의 시뮬레이션 측정을 수행함으로써 디자인 단계에서 미리 성능 평가와 예측이 가능하며, 이는 최적의 적응 전략을 결정하는데 보다 효과적으로 이용된다. 또한, 결정된 적응을 수행한 후, 기술된 각 개체들의 행위 모델을 통해 수행된 적응 결과가 기존의 기능이나 성능적 측면에서의 특성들을 지속적으로 만족하는지에 대한 검증을 통해 시스템의 일관성이 유지, 보장된다.

3. Resource Model

시스템이 동작하는데 영향을 미칠 수 있는 플랫폼 의존적인 모든 요소들을 표현하는 모델이다. 정적인 측면과 동적인 측면의 리소스로 분류하며, 기술하는 요소로 다음과 같은 요소들이 있다.

(표 1) 리소스 모델의 Property

	미리 정의되는 요소	실행 중 수집되는 요소	실제 수집된 정보
정적(Static)	Device (Type)	Device(Type)	MobileDevide (PDA)
	Os(Type)	Os(Type)	PPC(Pocket PC)
	Display (Size, Resolution)	Display (Size, Resolution)	320*240
동적 (Dynamci)	Port(Number)	Port(Number)	2000
	CPU(type,Total_size)	CPU(Type, Current_Usage)	(Intel Xscale, 640Mhz)
	Memory (Total_size)	Memory (Current_Usage)	(512M, 140M)
	Power (Total Size)	Power(MaxUpTime,remainUpTime, BatteryLevel)	(240mins 144mins, 60%)

정적 리소스는 디바이스의 타입이나 운영체제와 같은 동적인 변경이 적은 리소스들을 포함한다. 동적인 리소스는 CPU와 메모리의 사용량과 같이 지속적으로 변화하며 시스템에 영향을 미칠 수 있는 리소스들에 대한 정보를 정의하고 기술한 모델이다. 이들을 기술하기 위해 클래스 다이어그램을 이용한다.

4. Environment Model

환경 모델은 시스템 동작에 영향을 미칠 수 있는 시스템 주변의 요소로써 플랫폼 독립적인 모든 요소들을 표현하며, 네트워크 모델과 사용자

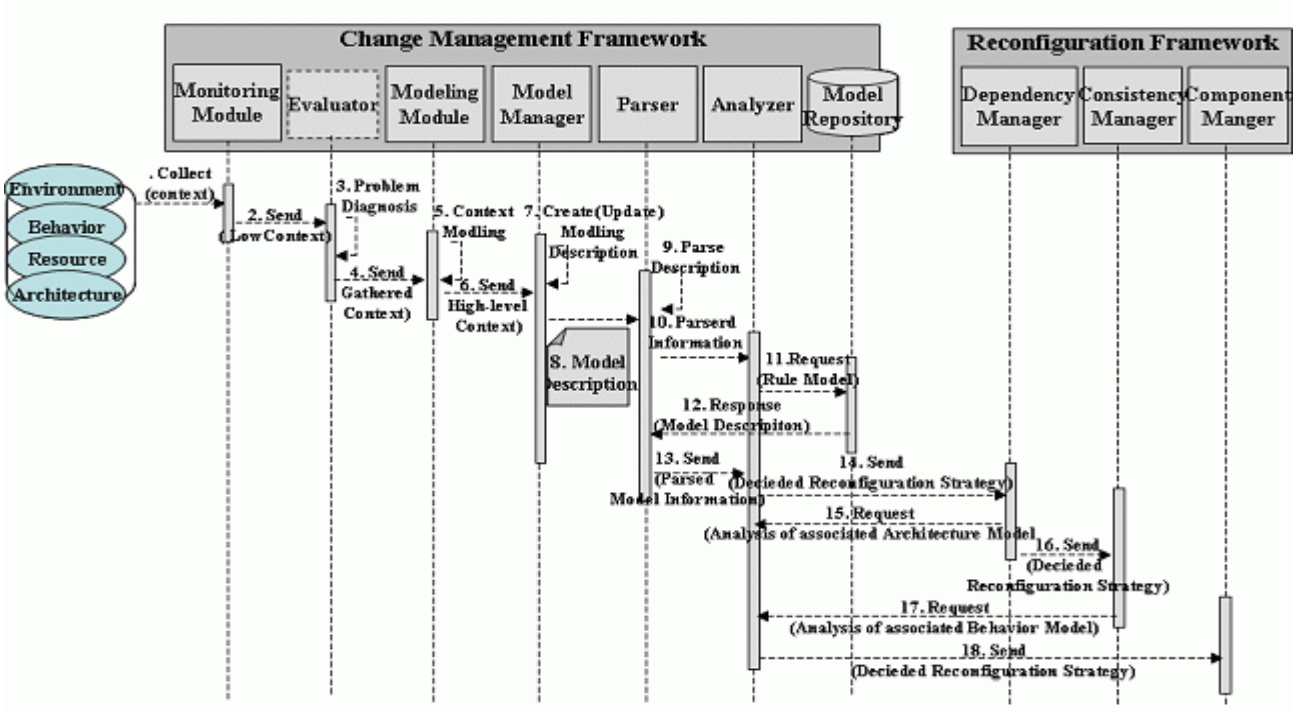
모델로 분류된다. 시스템과 상호 운용되며 영향을 미치는 현재 사용하는 통신 채널의 이용할 수 있는 밴드위스, 현재 이용하는 프로토콜 타입, 실행 중인 외부 시스템의 위치 주소, 사용 프로토콜과, 시스템 주변의 게이트웨이 및 미들웨어 정보의 위치 주소와 같은 정보들이 네트워크 모델에 포함된다. 사용자 모델에는 사용자의 피드백으로 얻어진 정보를 이용하는 사용자 만족도, 사용자의 히스토리 정보를 수집한 사용자의 선호도가 있다.

제시한 네 가지의 모델을 이용하여 Change Management Framwork 는 (그림 1)과 같이 동작한다.

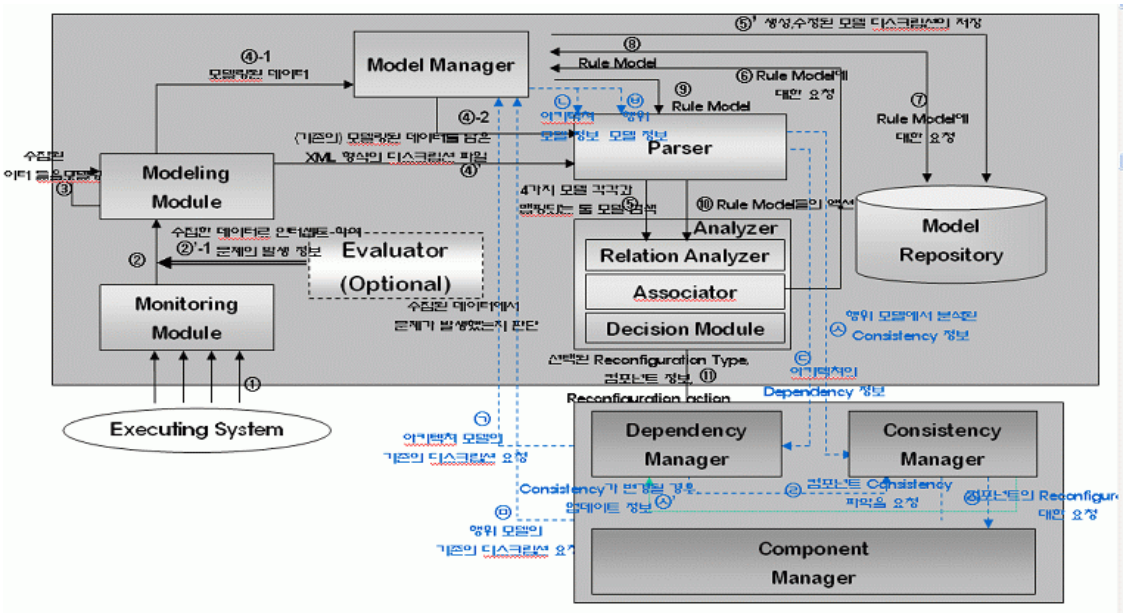
실행 중인 시스템은 정의된 모델을 기반으로 하여 지속적으로 모니터링을 수행하여 시스템의 아키텍쳐, 상태, 리소스들의 요소들을 관찰하고 수집한다. 수집된 데이터들은 모델에 맞는 적절한 값으로 변경되기 위한 정제 과정을 거치게 된다. 정제된 과정을 거친 추상화된 고수준의 데이터를 기반으로 하여 기존의 정의된 모델의 제약 사항 정보와 비교를 통하여 다른 좋지 않은 상황을 발견하는 분석과정을 거친다. 문제가 발생되었을 경우, 시스템은 미리 분석하고 정의된 모델의 적응 전략을 이용하여 시스템의 상태를 최적으로 유지하기 위해 적용할 적응 정책을 선택한다. 선택된 적응 정책을 실행 중인 시스템에 적용시킴으로써 컴포넌트의 재구성, 적응, 배치나 리소스의 사용량에 대한 조절을 통하여 시스템은 외부적 변화에 적절하게 적응하게 된다. 앞에서 언급한 네 가지 모델은 미리 정의되어 모니터링에서 적응까지의 모든 단계에서 유기적으로 상호 운용된다. 또한, 도메인 특성에 따라 추가되거나 조절이 가능하다.

3-2. 제안 시스템 프레임워크

이 장에서는 제안 시스템의 전체적인 아키텍쳐를 기술한다. 제안된 모델 기반의 시스템을 사용함으로써 어떻게 모델 기반의 변경 관리 프레임워크는 위의 언급한 네 가지 모델을 기반으로 하여 실행 중인 시스템 자신과 내부적, 외부 환경의 정보들을 수집하여 변경이 발생하거나, 문제가 발생된 부분에 대하여 분석을 수행하고, 적응하는 과정을 수행하는 프레임워크로 다음과 같은 모듈들로 구성된다. 이 프레임워크에서 분석된 적응 정보는 실제 적응을 실행하는 (그림 2) 하단의 Reconfiguration framework 와 연결되어 실행 중인 시스템에 실행된다.



(그림 1) 제안 시스템의 프로세스



(그림 2) 제안 시스템 아키텍처

시스템을 구성하는 모듈은 다음과 같다.

1. Modeling Module

수집된 저수준의 데이터들을 전송받아 미리 정의된 모델에 따라 정해진 Formal 한 메소드나 동작을 이용하여 고수준의 데이터로 변경되거나, 불필요한 값들은 필터링 하는 모델링 작업을 한다.

2. Evaluator (Optional)

시스템 도메인의 특성에 따라 선택적으로 적용할 수 있는 oprional 모듈로써, Modeling Module 에 주기적으로 전송되는 수집된 데이터들을 인터셉트하여 미리 정의된 제약 조건의 결과에 따라 구조, 환경, 행위, 리소스 중 어떠한 모델에 대한 변경이 필요한지 결정한다. 정의된 제약 조건을 위반하거나, 긴박한 처리가 요구될 때 사용된다.

3. Parser

Parser 에서는 각 모델의 디스크립션에 대한 해석 작업을 수행한다. 해석된 정보를 이용하여 Relation Analyser 는 룰 모델과의 맵핑을 수행함으로써 현재 해석된 모델을 구성하는 요소들로부터 영향을 받을 수 있는 다른 모델은 무엇이 있는가를 파악한다. 예를 들어, 해석된 모델이 대역폭 정보를 담고 있는 환경 모델일 경우, 이에 해당하는 룰 모델을 통해 그 외 구조, 행위, 리소스 세 가지의 모델과의 연관성 분석을 수행한다. 이를 통해 최종적으로 적응 정책에서는 가장 연관성을 갖는 리소스 모델의 개체 정보를 얻는다.

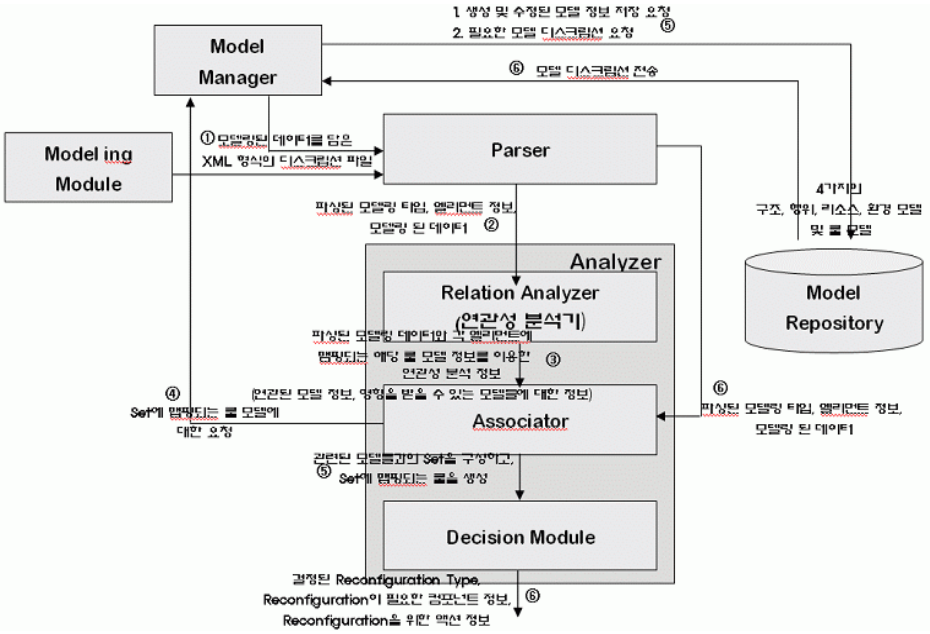
```
- Rule 1
If(Bandwidth<=MaximUM)Associate(ResourceModel.CPU);
else(Bandwidth>Minimum)Associate(EnvirtModel.Usersatisfication);
……..
- Rule 2
If(currentUsageCPU>80%)TransformReformatinge(GCIF);
If(currentUsageCPU>30%)TransformResolution(HIGH);
```

(그림 3) 모델의 연관성을 위한 Rule Model

모델링 모듈을 거쳐서 생성된 고수준의 데이터는 Model Manager 에서 전송 받아 제시한 기술 기법을 이용하여 XML 형식의 디스크립션으로 변경을 수행한다. 또한, 모델 매니저는 모델에 대한 생성과, 모델에 대한 수정과 확장 및 관리를 수행한다.

4. Analyzer

Analyzer 모듈은 Parser, Relation Analyzer, Associator, Decision Module 로 구성되어 있다.



(그림 4) Analyzer 의 프로세스

(그림 4)는 각 모듈을 이용한 동작 흐름은 기술한다. 전송된 정보를 이용하여 Associator는 연관된 모델들에 대한 요소를 이용하여 일련의 집합을 구성하여 해당하는 룰 모델을 요청하고, 모델 저장소를 통해서 미리 정의된 룰 모델을 가져온다. Decision Module을 통하여 모델들의 집합을 가지고 재구성을 위해 기능적, 비기능적, 구조적 적응 중 타입을 판단하고, 결정된 재구성의 타입과 관련된 컴포넌트 정보들을 Dependency Manager에 전송한다.

5. Reconfiguration Framework

재구성 할 컴포넌트에 대하여 어떠한 컴포넌트들이 연관되어있는가, 그리고 컴포넌트를 변경했을 영향을 받는 관계를 가지고 있는 다른 컴포넌트들에 대한 의존성를 위해 아키텍쳐 모델을 요청하고, Parser 를 통해 아키텍쳐 모델에서의 의존성 정보를 가져온다. 컴포넌트를 변경했을 때 일관성을 판단하기 위해 행위 모델을 가져오고 행위모델을 통해 일관성을 판단한다. 재구성에 대한 일관성 정보를 분석 후, 최종적으로 모델을 기반의 재구성 정보를 이용하여 Component Manager 는 적응을 수행한다.

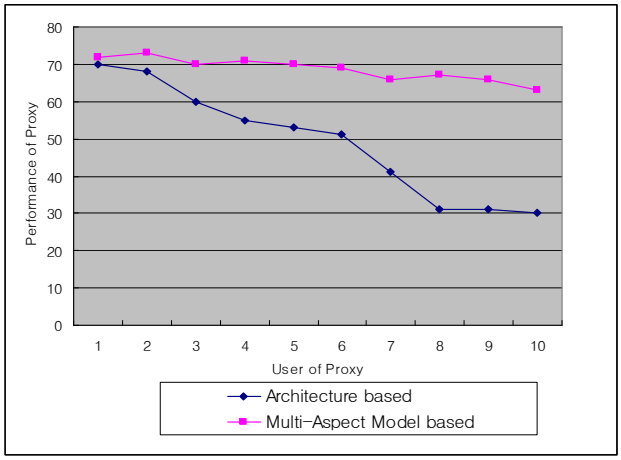
4. 적용 사례 및 평가

이 장에서는 제안 시스템의 평가를 위해 프로토타입을 구현하고, 원격화상회의시스템에 적용하여 적응의 정확성과 효율성을 측정하였다. (표 2)는 제시한 네 가지 모델을 기반 시스템의 적응의 효율성을 정리한 표이다.

(표 2) 고려하는 컨텍스트 정보

	다중 모델 기반의 장점
아키텍쳐 모델	효과적인 컴포넌트의 재구성이 보장과 이를 통한 시스템의 정제, 진화
행위 모델	1. 디자인 단계에서 성능 평가의 측정으로 적응의 효율성의 증가 2. 적응 후, 기존의 기능이나 성능에 대한 일관성의 유지 및 보장이 가능
리소스 모델	보다 다양한 자원들의 기술과 다른 모델에 미치는 영향에 대한 연관성을 제공
환경 모델	폭넓은 네트워크 요소와 사용자 요소의 기술 및 관리

다음은 제안 시스템을 원격화상시스템의 적용하여 적응의 효율성을 확인한 결과이다. 원격화상시스템에서 일반 PC 사용자 2 명과 모바일 프록시, 모바일 프록시에 접속한 모바일 클라이언트 사용자가 회의를 진행하는 경우, 프록시에 모바일 디바이스를 가진 사용자들이 지속적으로 회의 접속하였을 경우를 측정하였으며, (그림 5)는 프록시의 퍼포먼스에 대한 실험 결과를 나타낸 그래프이다.



(그림 5) 모바일 프록시의 Performance

기존의 아키텍처 기반 시스템에서는 아키텍처 모델 내의 모바일 클라이언트 컴포넌트 수가 증가함에 따라 모바일 프록시와 모바일 클라이언트의 아키텍처의 재구성만을 반영하여 적응을 수행한다. 그러나, 네 가지 모델 기반의 제안 시스템에서는 아키텍처 모델과 그외의 모델간의 연관성을 통해 아키텍처가 재구성됨으로써 모바일 프록시의 퍼포먼스와 리소스 모델의 각 요소에 미치는 영향을 진단하고, 행위 모델을 통해 효과적으로 퍼포먼스를 유지하기 위해 시스템의 행위들을 추가 및 변경함으로써, 적응을 수행한 후에도 모바일 프록시의 퍼포먼스가 지속적으로 유지되는 것을 볼 수 있었다.

5. 결론

본 논문에서는 구조, 행위, 리소스, 환경 네가지의 다양한 모델을 제시하고, 이를 이용하여 효율적인 적응을 위한 적응 프레임워크를 설계하고 제안하였다. 또한, 제안 시스템의 프로토타입 시스템을 구현함으로써 실험을 통해 제안한 프레임워크의 타당성을 입증하였다.

아키텍처 모델을 통해 이질적인 컴포넌트들을 유연하게 재구성 및 조립할 수 있는 적응과 동시에 시간 흐름에 따라 시스템의 다양한 활동들을 기술한 행위 모델을 이용하여 시스템의 새로운 행위를 추가하거나 이동함으로써 동적 적응까지 수행할 수 있다. 기술된 행위 모델들의 성능 평가 요소들을 통해 다양한 적응 전략에 대해 성능의 시뮬레이션 측정을 수행함으로써 디자인 단계에서 미리 성능 평가와 예측이 가능하며, 이는 최적의 적응 전략을 결정을 제공한다. 또한, 결정된 적응 전략을 수행한 후, 기술된 각 개체들의 행위 모델을 통해 수행된 적응 결과가 기존의 기능이나 성능적 측면에서의 특성들을 지속적으로 만족하는지에 대한 검증을 통해 시스템의 일관성이 유지, 보장할 수 있다. 시스템과 외부 환경에 문제가 발생했을 경우, 기존의 아키텍처 모델 기반 시스템보다 제시한 다양한 모델간의 상호 연관성을 통해 다각도에서 보다 정확하고 체계적인 진단과 적응 후의 시스템 퍼포먼스까지 고려가 가능하다.

6. 참고 문헌

[1] R. Laddaga, "Active Software",IWSAS2000,LNCS 1936,Springer-Verlag,2000,Pages:11‐26,17-19 April 2000.

[2] Peyman Oreizy, etc. al. "An Architecture-Based Approach to Self-Adaptive Software" IEEE Educational Activities Department, Volume 14, Pages: 54 - 62,May 1999.

[3] Quianxian Wang, etc al., CIMPSAC'03 IEEE "Runtime Software Architecture Based Software Online Evolution"

[4] Karsai, G.; Sztipanovits, J; "Model Based Approach To Self-Adaptive Software", Intelligent Systems and Their Applications, IEEE Volume 14, Issue 3, May-June 1999 Page(s):46 - 53

[5] Self adaptive software, DARPA, BAA 98-12, Proper Information Pamphlet, 1997. URL:http://www.darpa.mil/ito/Solicitations/PIP_9812.html.