가상 자원 및 서비스 관리를 위한 상황 인지 정보 모델

최영락 O,1, 리건 2, 한윤선 1, 홍원기 1,2

포항공과대학교 1 정보전자융합공학부, 2 컴퓨터공학과

{dkby, gunine, seon054, jwkhong}@postech.ac.kr

요 약

보다 새롭고 개인화된 서비스에 대한 수요가 점차 증가하는 가운데, 이를 지원하기 위한 시스템에서는 한정된 자원을 서로 공유하면서 필요로 하는 서비스에 할당하기 위해 매우 복잡한 연산을 수행한다. 이 때, 가상화 기술은 물리 자원의 추상화를 통해 복잡한 연산을 간략화하는 데 도움을 준다. 뿐만 아니라, 가상 자원 및 서비스 관리를 지원하는 오토노믹 시스템은 서로 다른 서비스들에 대한 상황(Context)을 고려하여 특정 서비스의 가상 및 비가상 자원의 수요를 동시에 만족시킬 수 있는 중요한 관리적인 이점을 제공한다. 본 논문은 오토노믹 시스템에서 가상 자원 및 서비스를 관리하기 위한 요구 사항들을 프로비저닝 관점에서 기술하고, 오토노믹 시스템을 구성하기 위해 필요한 상황 인지 정보 모델을 설명한다. 그리고 해당 정보 모델을 클라우드 컴퓨팅에서 SLA 를 관리하기 위한 시나리오에 적용하여 적용 가능성을 살펴본다.

1. 서론

다양한 기술이 생겨나고 각 장치들이 점차 복잡해지면서 이들로 구성된 IT 시스템을 관리하기 보다 어려워지고 있다. 특히, 서로 다른 요구 사항 및 비즈니스 목표를 가진 여러 서비스를 제공하기 위한 시스템은 다양한 비즈니스적인 요소를 동시에 만족시켜야 하므로 많은 복잡성이 요구된다. 이러한 복잡성은 네트워크 및 서버 인프라에 있어 같은 자원들을 공유하는 각 서비스들의 다양한 요구 사항들을 어떻게 수용 가능한 지, 그리고 점차 많아지고 다양해지는 관리 및 운용에 사용되는 정보들을 시스템이 어떻게 수용 가능한 지로 나눌 수 있다. 이 때 개인화된 서비스를 시스템에 적용하는 경우, 특정 사용자의 요구 사항을 시스템에서 고려하여 사용자들의 보다 많은 요구 사항들을 충족시킬 수 있다. 그러나, 해당 서비스가 서로 다른 사용자들에게 최적화되어 전달될 수 있도록 서비스를 변화시켜야 할 뿐만 아니라, 이러한 변화에 대응하여 관리되어야 하므로 시스템은 점차 복잡해질 수밖에 없다.

이러한 복잡성을 해결하기 위한 방안 중 하나로, 가상 자원들을 활용하여 개인화된 서비스를 제공하는 방법이 있다. 가상화는 서로 다른 유지 보수 및 보안 요구 사항을 가진 각 머신 및 어플리케이션들의 하드웨어, 운영 체제, 어플리케이션 의존성을 낮추어 서로 격리시키는 방식을 통해 관리적인 이점을 제공한다 [11, 12]. 서로 다른 서비스들이 사용하는 물리적인 서버와 같은 자원들을 가상화시켜 동적으로 할당함으로써 각 서비스의 비즈니스적인 요구 사항을 충족시킬 수 있지만, 이러한 가상화 기술은 매우 주의 깊게 관리되어야 한다. 가상화 기술을 효과적으로 이용하기 위해서 특정 서비스가 어떤 가상 자원을 할당 받을 것인지를 설정하는 데 있어 정책 기반 관리가 이용될 수 있다.

제안하는 모델링 접근 방식은 개인화된 서비스를 제공하기 위해 가상 자원 관리 방식와 오토노믹 시스템 아키텍처 간의 공백을 채우는 데 목적을 두고 있다. 개인화된 서비스를 제공하는 다양한 어플리케이션들이 오토노믹 아키텍처 상에서 실행될 때, 가상화 기술은 어플리케이션들이 활용 가능한 자원들을 찾는 과정을 간략화할 수 있다. 클라우드 컴퓨팅에서, IaaS(Infrastructure as a Service)는 주어진 서비스 별로 묶인 가상 자원 집합에 대한 접근성을 제공하는데, 이 가상 자원 집합은 개인화된 서비스를 구현 시 활용 가능한 빌딩 블록을 형성하는 데 이용될 수 있다 [13].

본 논문은 가상 자원 및 이기종의 비가상 자원, 그리고 해당 자원들로 이루어진 서비스들을 관리할 수 있도록 DEN-ng 정보 모델 [1]을 확장하는 데 초점을 두고 있다. DEN-ng 는 관리 환경에 존재하는 다양한 요소들을 나타내기 위한 ACF (Autonomic Communication Forum)에서 정의한 객체 지향의 표준 정보 모델로, 5 장에서 타 정보모델과 비교하여 설명하고자 한다. 제안하는 모델을 활용하여 가상 및 비가상 자원들을 프로비저닝할 때 해당 자원들을 설정 및 구성하는 노력을 줄이고 서비스를 정의, 배포 및 관리하는 절차를 간략화할 수 있다. 또한, 제안하는 모델은 가상 및 비가상 자원들을 동시에 통합된 관리 방식을 활용하여 어떻게 관리 복잡성을 줄일 수 있는지를 설명한다.

본 논문의 구성은 다음과 같다. 2 장에서는 개인화된 서비스를 제공하기 위해 가상 자원 및 서비스

본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업(정보통신)의 일환[KI0035594, HiMang(Highly Manageable Network and Service Architecture for New Generation) 원천 연구] 및 한국연구재단을 통해 교육과학기술부의 세계수준의 연구중심대학육성사업(WCU)으로부터 지원받아 수행되었습니다. (R31-2010-000-10100-0)

들을 관리하는 데 있어 핵심이 되는 모델링 요구 사항들을 프로비저닝 관점에서 정리한다. 3 장에서는 DEN-ng 7.0 모델을 확장한 주요 결과를 설명한다. 4 장에서는 제안한 모델에 대한 적용성을 설명한 Use Case 를 보여준다. 5 장에서는 관련 연구를 기술하고 마지막 6 장에서는 결론 및 향후 연구에 대해 기술한다.

2. 모델링 요구 사항

본 장에서는 가상 자원 기반의 개인화된 서비스를 제공하는 오토노믹 시스템을 관리하기 위한 정보 모델의 요구사항을 프로비저닝 관점에서 기술한다. 본 논문에서는 프로비저닝을 '특정 SLA (Service Level Agreement, 서비스 수준 계약)를 만족하는 서비스(또는 자원)들을 사용자에게 전달하는 일련의 과정'으로 정의한다.

ITIL (Information Technology Infrastructure Library) [2]와 eTOM (enhanced Telecom Operations Map) [3]에서는 프로비저닝 및 관련 기능들을 포함하는 워크플로우를 처리하는 과정을 추상적인 단계에서만 정의하고, 구체적인 인프라 컴포넌트의 프로비저닝 방법에 대해서는 기술하지 않고 있다. 본 논문은 관리 관점에서 상위 단계의 비즈니스 및 워크플로우 처리 과정을 포괄하며, 비즈니스 개념과 SLA 등의 상위 단계부터 프로토콜 장치 설정과 같은 네트워크 수준까지 존재하는 일련의 공통 객체들과 그들 사이의 관계를 정의하여 세부 내용을 구체화하고자 하였다. OMG 가 제안한 Model Driven Architecture (MDA) initiative [4]는 프로비저닝 과정에 대해 생성, 배포 및 관리에 필요로 하는 여러 객체들에 대한 코드를 생성하기 위해 사용된다. 이러한 관점에서 고안된 정보 모델이 반드시 만족해야 하는 핵심 요구 사항들을 다음과 같이 나열하였다.

정보 모델은 엔티티들을 확장이 용이하도록 설계되어야 한다. 이를 만족하기 위해 Role-object pattern [6] 및 Policy rule pattern [1] 등과 같은 재사용 가능한 소프트웨어 패턴 [5]들을 활용하여, 엔티티들 및 각 엔티티로부터 파생된 엔티티들을 표현 가능하다. 또한, 소프트웨어 패턴의 사용을 통해 클래스 수의 급증을 막을 수 있는 확장 가능한 프레임워크를 제공할 수 있다.

정보 모델은 엔티티와 메타데이터를 명확히 구분할 수 있어야 한다. 메타데이터는 관리 대상인 엔티티의 다양한 면 및 작용 상태를 기술하기 위한 데이터로, 엔티티와 구별되어 정보 모델에 표현되어야 한다.

정보 모델은 시스템 동작을 관리하는 다양한 정책 규칙을 나타낼 수 있어야 한다. 일반적으로, 주어진 특정 상황에서 많은 수의 정책 규칙들이 동시에 적용될 수 있다. 따라서, 시스템의 동작 상태 및 서비스를 명확히 정의하기 위해 주어진 상황에 해당하는 정책 규칙을 명확히 서술하는 것은 매우 중요하다. 이를 지원하기 위해, 상황 정보를 활용하여 주어진 지점 및 환경에 적용 가능한 정책 규칙 집합을 선택하는 과정 및 관련된 엔티티들을 모델링하고자 하였다.

3. 확장된 DEN-ng 정보 모델 제안

본 장에서는 자원 및 서비스들의 프로비저닝 및 관리 관점에서, 보다 유연하고 풍부한 특성을 지니도록 DEN-ng 정보 모델 7.0 을 확장하고자 한다. 또한, 가상화 기술을 적용하고 관리하기 위하여 DEN-ng Resource 도메인에 초점을 맞춘 확장 방식에 대해 구체적으로 기술한다.

3.1. 가상 자원 모델링

그림 1 은 DEN-ng 7.0 정보 모델 중 자원에 해당하는 Resource 계층을 나타낸다. DEN-ng 7.0 에서는 자원에 해당하는 Resource 클래스를 "관리 대상인 환경 내에서 관심을 갖고자 하는 물리적인 또는 논리적인 엔티티"로 정의하고, Resource 클래스에 3 개의 하위 클래스인 PhysicalResource(예: 만질 수 있는 객체), LogicalResource(예: 만질 수 없는 객체), 그리고 자원의 물리적인 특성과 논리적인 특성을 동시에 지니는 CompoundResource(예: 물리적/논리적 컴포넌트들로 이루어진 토폴로지 다이어그램에서의 한 클래스)들을 두었다.

가상 자원의 Semantic 을 지원하도록 모델링하기 위해, 우리는 먼저 가상 자원 클래스에 해당하는 VirtualResource 를 정의하였다. "VirtualResource 는 논리적인 작업으로부터 물리적인 현상을 분리시킨 추상화한 엔티티를 의미한다." 해당 정의는 AUTOI FP7 팀[7]에서 제안한 VirtualResource 정의와는 차이가 있으며, 이에 대한 자세한 설명은 5 장에서 설명하고자 한다.

그림 2 는 DEN-ng 7.0 정보 모델에서 가상 자원을 고려하기 위해 자원에 해당하는 Resource 계층을 수정한 결과를 나타낸다. 현존하는 DEN-ng 모델과의 호환성을 유지하기 위해 DEN-ng Resource 계층에 변화를 그림 2 와 같이 주었으며, 이 변화를 수용하기 위해 기존 각 클래스의 정의를 바꾸었다. 첫째, 우리는 Resource 클래스 계층에 VirtualResource 와 NonVirtualResource 를 포함하도록 하위 클래스를 바꾸었다. 둘째, 우리는 Resource 클래스 정의를 위 2 개의 클래스를 포함하도록 변경하였다. 따라서 "Resource 는 가상화된 또는 가상화되지 않는 컴포넌트 또는 시스템을 말한다. Resource 는 관리 대상인 환경 내에서 물리적인 또는 가상화된 엔티티들을 포함한다. Resource 는 다른 엔티티를 필요로 하는 제한되거나 중요한 엔티티를 표현할 수 있다."와 같이 Resource 클래스 정의를 변경하였다. 셋째, 우리는 기존 DEN-ng 에서 사용되었던 Resource 정의를 NonVirtualResource 의 정의로 사용하였고 원래

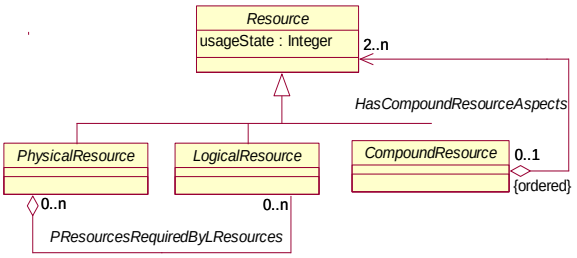

그림 1. DEN-ng 7.0 Resource 계층

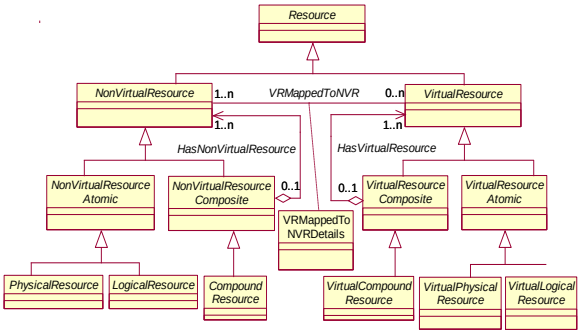

그림 2. 가상 자원을 고려한 수정된 Resource 계층

Resource 계층을 NonVirtualResource 계층으로 변경하였다. 마지막으로, 계층의 단순화를 위해 CompoundResource 계층을 제외하고 각 Resesource 하위 클래스에서 파생되도록 하였다.

NonVirtualResource 와 VirtualResource 간의 연관 관계(association)를 추가하여 VirtualResource 가 여러 개의 NonVirtualResource 에서 동작할 때의 의존성(예: 특정 물리적인 호스트에서 실행되는 경우)을 모델링 하였다. 그리고 Composite pattern [1]을 VirtualResource 및 NonVirtualResource 계층에 적용하여 확장성 및 재사용성을 높이고자 하였다. CompoundResource 는 물리적인 특성과 논리적인 특성을 동시에 지닌 NonVirtualResource 에 해당하므로 NonVirtualResourceComposite 의 하위 클래스로 두었다.

3.2. 상황 인지 정책 규칙에 의한 관리 모델링

그림 2 의 정보 모델을 기반으로 가상 자원들을 관리하는 경우, 가상 자원과 비 가상자원간의 복잡한 관계는 NonVirtualResource 클래스와 VirtualResource 클래스 사이의 VRMappedToNVR 연관 관계를 통해 각 자원간의 의존성을 관리할 수 있다. 그러나, 이 연관 관계와 관련하여 나타나는 Semantic 을 모델링 하기 위해서는 관련된 추가적인 클래스 및 관계를 정의해야 한다. 이 때. 적절한 소프트웨어 패턴을 사용하지 않고 모델링을 할 경우, 신축성에 대한 문제를 다루기 힘들어진다. 이를 해결하기 위해 Policy pattern [1]을 적용하였다.

DEN-ng 에서는 상황 정보를 모델링하기 위해

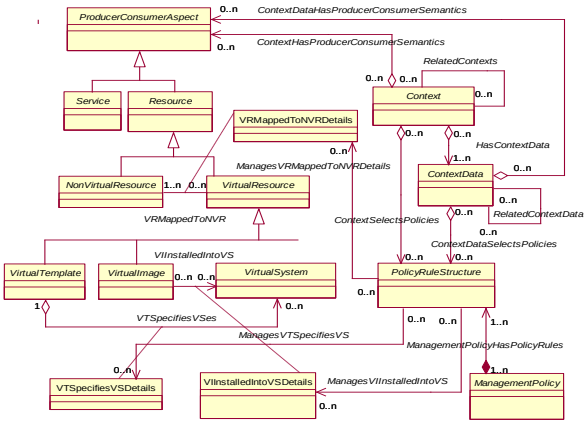

그림 3. 상황 인지에 따른 정책 규칙 기반의 관리 모델링

상황 및 상황의 변화를 Context 및 ContextData 라는 2 개의 상위 클래스로 모델링 하였다. ContextData 는 상황의 서로 다른 측면을 표현하는 데 반해, Context 는 어떤 상황을 완벽히 표현하는 완전한 집합체를 나타낸다. 예를 들어, 어떤 가상 서비스가 서로 다른 가상 장치를 이용하여 구성되는 경우, 각 가상 장치가 제공하는 자원(Resource)들은 ContextData 에 해당하고, 가상 서비스는 Context 로 모델링 될 수 있다. DEN-ng 에 표현된 모든 Resource 및 Service 클래스들은 Context 객체와 연관을 맺어 상황의 변화를 모니터링한다.

그림 3 은 상황 객체를 이용하여 VirtualResource 하위 클래스들을 정책 기반으로 관리하기 위한 모델링을 나타낸다. Policy pattern 은 일반적으로 적용되는 다양한 Anti-pattern 들을 사용하지 않도록 공유되거나 재사용 불가능한 정책 규칙(Policy rule)을 모델링 한다. 어떻게 VirtualResource 들이 생성되고 관리되는지에 대한 다양한 정책 규칙들이 적용될 수 있기에, 디자인 단계에서 문제가 발생하였을 때 재사용 가능한 해결책을 제시하는 Policy pattern 은 위와 같은 특징들을 모델링 하기 위해 매우 유용한 방식이라 할 수 있다. Policy pattern 을 적용함으로써, 정책을 모델링함에 있어 해당 Content 와 독립적으로 나타내었으며, 정책의 특성에 따라 해당 연관 관계의 semantic 도 변경할 수 있다. 이러한 Policy pattern 은 PolicyRuleStructure 와 VRMappedToNVRDetails 간의 연관 관계를 추가하고, Policy 와 관련된 클래스들을 이용하여 정책 규칙들을 보다 확장 가능한 방식으로 정의 할 수 있었다. DEN-ng 에서 Policy 와 관련된 대표적인 클래스들로는 PolicyRuleStructure (Event-condition-action, Goal, Utility function policy 등의 여러 종류에 대한 정책 규칙을 모두 포함하는 상위 클래스 [1])과 ManagementPolicy(특정 대상 엔티티에 한정된 Policy rule 에 대한 상위 클래스 [1]) 가 있다. 예를 들어, 서로 다른 메타데이터는 다른 종류의 정책 규칙에 사용되어 각 정책 규칙이 특정 종류의 관리 대상 엔티티에 적용됨을 설명할 수 있다.

정책 규칙은 상황의 변화에 따라 지배하는 방식이 변경되므로 PolicyRuleStructure 클래스는 Context 클래스와 연관 관계를 맺는다. DEN-ng 에서는 상황과 관련된 Context 와 ContextData 클래스를 이용하여 ProducerConsumerAspect 하위 모든 클래스들의 상황을 감시하고, 이에 따른 적절한 정책 규칙들의 집합을 선정한다. 또한 이 과정을 통하여 필요한 Operation 들을 관리 규칙(ManagementPolicy 클래스)에 해당하는 정책 규칙(PolicyRuleStruture 클래스)에 따라 동작시키도록 한다. 정책 규칙은 엔티티를 다루는 역할을 선택하고, 이는 기능성 및 Control loop 에 의해 모니터링 되는 관리 및 운영 데이터들을 정의하는 데 이용된다. 따라서, 관리되고 있는 엔티티의 상태 변화를 통해 상황 변화를 감지할 수 있으며, 이는 임의의 시간에 주어진 서비스 및 자원들을 정의하는 정책 규칙들을 조절하는 데 사용된다.

4. 가상 자원 기반의 클라우드 컴퓨팅 관리 적용 가능성

클라우드 컴퓨팅 환경에서 가상 자원을 기반으로 이루어진 개인화된 서비스가 신축성에 따라 변화하는 SLA 를 수용하는 Use Case 시나리오를 통해 제안된 모델의 적용 가능성을 보이려고 한다. 이 시나리오를 통하여 클라우드 컴퓨팅 환경에서 계약 사항을 위배하지 않으면서 가상 자원들이 어떻게 관리되고 이를 활용하는 서비스들이 고객에게 전달될 수 있는지를 보여주고자 한다.

SLA 는 두 당사자(party)간에 상품, 서비스, 중요성, 책임 등에 관한 공통의 이해 사항을 명시하기 위해 고안된 나타내는 일종의 계약으로, 개인화된 서비스에서 예상되는 세부 사항을 기술하는 방법으로 쓰일 수 있다. DEN-ng 에서 SLA 는 Quality of Service(QoS) 와 같은 구체적인 목표를 이루고 유지하기 위한 적절한 과정 및 대상 Metric 들을 구체화하지만, 개인화된 서비스에 대한 계약 사항 및 가상 자원 및 비가상 자원 모두의 특성을 반영하기에는 쉽지 않다. 몇몇 SLA 사항들은 구체적인 자원에 대한 언급을 필요로 하는데, 가상화 기술은 이기종 자원간의 물리적인 세부 사항을 숨기는 역할을 한다. 또한, SLA 용어들은 상황(Context)에 따라 다르게 해석될 수 있다. 예를 들어, 개인화된 서비스를 구성하기 위한 보안 수준을 기술하고자 할 때, 동일인이라 하더라도 처한 상황(예: 작업 중, 엔터테인먼트)에 따라 다른 보안 요구 사항을 가지며, 이와 관련된 다른 프라이버시 요구 사항을 갖기도 한다. 또한 클라우드 컴퓨팅 환경에서는 신축성의 특징에 의해 사용자가 자원을 필요로 하는 양 만큼만 자원을 할당하는 동적 프로비저닝을 필요로 하는데, 이러한 동적 프로비저닝에 영향을 주는 요소들은 상황에 따라 달라진다. 이러한 문제를 해결하기 위해, 본 모델링 접근 방식을 통하여 항목 및 기능들을 파악하고 이들을 가상 및 비가상 자원에 관련하여

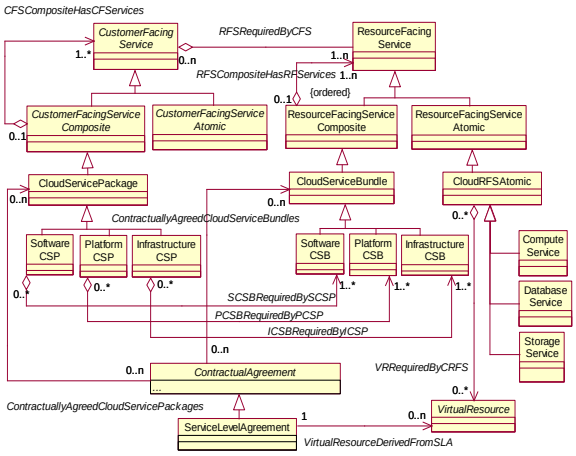

그림 4. 클라우드 컴퓨팅 환경에서 서비스, 가상 자원 및 SLA 간의 매핑

SLA 에 대한 Mapping 을 보다 효과적으로 돕고자 한다.

그림 4 는 클라우드 컴퓨팅 환경에서 SLA 변화에 따라 가상 자원이 프로비저닝되고 관리되는 매핑을 설명한다. 서비스 제공자는 클라우드 컴퓨팅 서비스를 제공하기 위해 해당 서비스를 정의하는데, DEN-ng 에서는 사용자 중심의 서비스인 CustomerFacingService 클래스와 실제 자원들을 관리하여 사용자 중심의 서비스를 지원하는 ResourceFacingService 클래스로 구분하였다. 해당 클래스들은 Composite Pattern 으로 설계하여 확장성을 지원한다. VirtualResource 가상 자원 클래스는 ServiceLevelAgreement 라는 SLA 클래스에 기술된 사항에 따라 프로비저닝 및 모니터링이 이루어진다.

ServiceLevelAgreement 의 서비스와 관련된 사항들은 ContractualAgreement 와 연관 관계를 맺는 CloudServicePackage 및 CloudServiceBundle 클래스들을 사용하여 기술되는데, CloudServiceBundle 클래스는 QoS 로 구체화 가능한 자원들과 Mapping 되므로 ResourceFacingService 의 하위 클래스에 해당한다. 반면, CloudServicePackage 클래스는 주어진 SLA 에 따라 다루어져야 하는 일련의 어플리케이션들을 포함하고 있으며, CloudServicePackage 내 각 어플리케이션에 Mapping 되는 일련의 CloudServiceBundle 들을 이용하여 어플리케이션을 전달한다. 이러한 Relationship 을 통해 서비스 제공자와 고객 사이의 계약이 바뀌더라도 가상 자원을 기반으로 한 개인화된 서비스의 프로비저닝이 보다 강화된다. 비즈니스 요구사항 또는 고객의 요구의 변화에 따라 SLA 의 수정이 발생하여도, SLA 와 관련된 클래스들의 연관 관계들을 통해 계약을 이행하기 위해 변화되어야 하는 관련된 클래스들을 찾을 수 있다. 또한 이러한 변화는 Context 및 ContextData 클래스에서 해당 상황을 인지하여 그림 3 에서 설명한 정책 기반의 관리를 제공할 수 있도록 한다.

5. 관련 연구

[1]에서는 DEN-ng 정보 모델을 DMTF CIM [9] 모델 및 TMF SID [10] 모델과 비교하였다. 본 논문에서 DEN-ng 를 선택한 주요 이유로는 다음과 같다. 첫째, CIM 과 SID 는 상황 모델이 없는 반면, DEN-ng 는 잘 설계된 상황 모델을 포함하여, 필요로 하는 가상 자원 및 서비스들을 상황에 따라 결정하고, 적용 가능한 정책들을 선택하는 데 결정적인 역할을 수행한다. 둘째, DEN-ng 는 메타데이터 모델을 포함하지만, CIM 과 SID 에는 존재하지 않는다. CIM 과 SID 는 그 자체의 특징을 기술한 데이터 및 관리 대상의 행위, 그리고 관리 대상인 엔티티가 어떻게 사용되는지를 기술하는 데이터를 구분하지 못하여 확장성을 제한한다. 셋째, DEN-ng 와 비교해 볼 때, CIM 은 소프트웨어 패턴을 사용하지 않았고, SID 는 일부분에만 적용하여 확장성이 낮다. 넷째, CIM 과 SID 는 DEN-ng 에서 제공하는 Finite State Machine 과 같은 구성과 관련된 행위를 나타내는 메커니즘을 제공하지 않아 State 와 Action 을 통해 코드를 모델로부터 생성하는 데 어려움이 있다. 마지막으로, DEN-ng 는 CIM 과 SID 와 달리 엔티티의 전반적인 모델링을 수행하였기에 한 엔티티의 여러 측면뿐만 아니라 엔티티 전체를 나타내고, 이는 특정 행위를 나타내고 관리하는 데 더욱 강력하고 확장 가능한 방식을 제공한다.

AUTOI 프로젝트 [7]는 네트워크 자원 및 서비스에 대한 가상화를 모델링 하였다. AUTOI 프로젝트는 비즈니스 목표와 네트워크 기능들을 Mapping 함으로써 가상화된 네트워크 자원 및 서비스들을 제어하기 위한 관리 오버레이를 정의하는 것을 목표로 한다. AUTOI 모델은 필요한 개념들을 캡쳐하고 표현하기 위한 공통 언어로써 확장시킨 DEN-ng 정보 모델을 사용하였다. 그러나 이 모델은 본 논문에서 기술한 모델에 비해 기능적인 설명과 확장성이 부족하다. 특히, AUTOI 모델에서 VirtualResource 클래스는 Resource 의 하위 클래스로 정의되고 있는데 이는 DEN-ng 의 Resource 클래스 정의에 해당하는 Resource 가 아니기에 기술적으로 맞지 않다. 반면, PhysicalResource 와 VirtualPhysicalResource, 그리고 LogicalResource 와 VirtualLogicalResource 간에 상관 관계를 맺고 있는데 이를 통해 가상화된 서비스를 프로비저닝 하기에는 충분하지 않다.

6. 결론

비록 여러 가상 자원 및 서비스들이 이미 배포되어 사용 중이지만, 이들을 관리하는 것은 여전히 매우 복잡한 과정을 필요로 한다. 현재까지, 가상 자원 및 가상화된 서비스를 프로비저닝하고 관리하는 데 사용 가능한 구체적인 모델 및 정보 모델을 기반으로 한 클라우드 컴퓨팅 관리 방안은 존재하지 않는다. 본 논문은 이러한 모델에 대해 개략적으로 기술하고 있으며, 어떻게 이러한 모델이 클라우드 컴퓨팅에서 개인화된 서비스에 이용 가능한지를 설명한다. 본 논문에서는 확장성을 제공하는 소프트웨어 패턴을 활용하여 클래스들과 관련 연관을 정의하는 과정을 통해 가상 자원 및 가상화된 서비스를 구체화하였다.

향후 과제로는, 이 모델을 발전시켜 어떻게 VirtualResource 및 VirtualService 가 온톨로지 및 First order logic 을 사용하여 관리되는지에 대한 프로토타입을 만들어보고자 한다. Fact 에 대한 모델에 상응하는 부분을 전달하고 First order logic 을 활용한 추론 가능한 온톨로지를 활용하여 데이터를 다룰 수 있게 된다. 이후, 이 모델 및 온톨로지를 오토노믹 아키텍처에 적용하여 클라우드 컴퓨팅 시나리오를 현실화하고, 어떻게 보다 개인화된 서비스들을 VirtualResource 와 VirtualService 클래스를 기반으로 프로비저닝 및 관리되는 데 이용 가능한지에 대한 프로토타입을 설계하고자 한다.

7. 참고 문헌

[1] J. Strassner, “Introduction to DEN-ng”, Tutorial for FP7 PanLab II Project, January 21, 2009.

[2] ITIL, http://www.itil-officialsite.com/home/home.asp.

[3] TMF, eTOM, http://tmforum.org/BusinessProcessFramework/1647/home.html.

[4] OMG, MDA, http://www.omg.org/mda/.

[5] Gamma, E., Helm, R., Vlissides, J., “Design Patterns-Elements of Reusable Object-Oriented Software”, Addison-Wesley, November 2000.

[6] D. Bäumer, D. Riehle, W. Siberski, M. Wulf, “Role Object”, In Proceedings of the 1997 Conference on Pattern Languages of Programs, 1997.

[7] AUTOI (Autonomic Internet, an FP7 project), available from: http://ist-autoi.eu/autoi/index.php.

[8] OpenNebula, available from: http://www.opennebula.org/documentation:documentation.

[9] DMTF, CIM Schema: Version 2.24, available from: http://www.dmtf.org/standards/cim/cim_schema_v2240.

[10] TMF, SID Schema, available from: http://tmforum.org/InformationFramework/1684/home.html.

[11] Federica (Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures) project, available from: http://www.fp7-federica.eu.

[12] Reservoir fp7 project, available from: http://reservoir-fp7.eu/index.php.

[13] IAAS framework project, available from: http://www.iaasframework.com.

[14] J. Strassner, Policy Based Network Management, Morgan Kaufman, San Francisco, 2003.