앙상블 아키텍처 기반 중앙관제 시스템 설계 방법에 대한 연구

조성원*, 김진오*,* 손광철***
*광운대학교 홀로그래피 3D 콘텐츠학과, **아주대학교 시스템공학과,
***광운대학교 정보콘텐츠학과
sone.cho@kgu.ac.kr, jokim1201@ajou.ac.kr, kcson@kw.ac.kr

A study on the design method of central control system based on ensemble architecture

Seung-Won Cho*, Jin-Oh Kim**, Kwang-Chul Son***
*Dept. of Holography 3D Contents, Kwangwoon University,
**Dept. of System Engineering, Ajou University,
***Dept. of Informatiom and Contents, Kwangwoon University

요 약

오늘날 정보화 시스템은 우리 생활에 아주 밀접하게 접근하고 있다. 모든 이들이 생활을 접함에 있어서 이제 정보화 시스템은 당연하게 여겨지고 오히려 그에 대한 중요성에 대해 망각하고 사용하게 되었다. 그러나 우리의 실생활에 있어서 접목되는 정보화 시스템들이 과연 적절하고 효율적으로 구성되었는지 충분히 고민을 해봐야 한다. 정보화 시스템은 기획되고 개발되고 사용되고 궁극적으로는 최적화되어야 한다. 본 논문은 I-MOD 시스템의 구조를 가지고 여러 아키텍처 설계 방법론 중 골조의 근간이 하나가 아닌 다양한 방법론의 장점들을 융합하여 다양한 시각 입장으로 설계 방법을 풀어내려 연구하였다.

1. 서론

1.1 연구 배경 및 목적

신규 전산 시스템을 구축하거나 기존 시스템을 통합하여 재구축하면서 어떻게 구축할 것인가 어떻게 해야 잘 만들 수 있을 것인가에 대한 고민은 계속되어왔다. 사용자의 요구사항에 따라 시스템의 구조를 설계하고 그 위에 기능들을 만들었으며 이것이 잘 만들어졌는지 여러 테스트 단계와 검증을 통해 우리는 시스템을 만들었다고 정의했다.

건물을 만들 때 건물의 근간이 되는 골조와 뼈대가 튼튼해야 안정성 있는 건물이 지어지는 것과 같이 시스템 또한 3A(Availability, Adaptability, Anti-hacking)의 완벽한 아키텍처 구성과 One 소스, One 통합 DB로 구축되어야 Application Program과 DB 아키텍처는 물론이고 UI/UX 환경에서 원활하고 생산적으로 운영되어 고객의 요구에 유연성 있게 대처하고 합목적으로 고객 또는 사용자들이 편리하고 쉽게 시스템을 사용할 수 있다.

아키텍처 방법론에 대해서 많은 연구기관과 기업들이 연구를 진행해왔다. Monolithic에서 SOA(Service Oriented Architecture)로 SOA는 또 MSA(Micro-service Architecture)로 방법론에 대한 발전과 이론적 배경은 이미 오래된 개념이다.

하지만 현실적으로 아직 산업에서는 초기 아키텍처 방법론인 Monolithic을 많이 사용하고 있다.

그러나 갈수록 시스템이 대형화되고 복잡해짐에 따라 아키텍처 방법론들의 장점을 결합하여 구성하는 것이 효율적일 것이라는 생각에서 연구하게 되었다.

이 연구는 3가지 아키텍처 방법론을 하나의 시스템에 적용하여 이에 대한 효과를 주장하기 위함이다. 이것을 토대로 많은 아키텍트 분들께 향후 새로운 아키텍처 방법론에 대한 아이디어 제공의 초석이 되었으면 한다.

1.2 연구 범위 및 내용

본 논문은 인천 스마트시티 챌린지 사업 영역의 서비스 중 하나인 i-MOD 시스템을 기준으로 연구하였다.

i-MOD(Incheon Mobility On Demand) 서비스란 버스를 대상으로 하는 수요반응 Mobility System을 말한다. 2019년 시범사업을 통해 2020년부터 인천 영종도를 비롯한 여러 산업단지를 경유 하여 사용자 필요에 의한 공유 버스 시스템을 도입 운영하고 있다. 본 시스템은 95% 이상이 Monolithic으로 구성되었고, 그중 일부 SOA를 적용하여 플랫폼화된 시스템이 운영되고 있다.

(그림 1) 전체 시스템 구조

지금도 여러 지자체에서 본 서비스를 도입하려고 준비 중이나, 서비스 지역에 특징을 가변적이고 유연하게 반영하기 위한 확장성과 특징 성이 부족함에 따라 본 연구에서 몇 가지 관점을 기준으로 앙상블 아키텍처 기법을 적용한 아키텍처 설계방법을 연구하여 제안하였다.

2. 아키텍처 방법론

본 논문에서는 Monolithic, SOA, MSA 3가지 아키텍처 방법론을 언급한다. 가장 오래되고 많이 사용되었던 Monolithic은 쉽게 개발환경을 구성하고 서비스 간의 연계성에 있어서 단순하게 구성할 수 있는 반면에, 시스템 규모가 커지면 복잡도 증가 및 배포에 시간이 많이 소요된다는 단점이 있다. SOA는 서비스 단위로 모듈을 설계하여 재사용성이 높고 Monolithic보다 유연하다는 장점이 있는 반면에 설계에 있어서 다소 높은 비용이 발생하고, 서비스 간의 일관성이 부족하다는 단점이 있다. MSA는 SOA가 진화된 것인데, 기능에 대한 유연성과 독립성이 매우 우수하나 테스트의 복잡도가 증대되고, 많은 비용이 소요된다. <표 1>은 3가지 아키텍처 방법론들의 특징을 표현하였다.

<표 1> 아키텍처 방법론에 대한 장단점 및 특징

구분	Monolithic	SOA	MSA
장점	• 배포, 테스트 관리가 용이하나, 새로운 기술 적용이 힘듦 • 로드밸런스를 이용한 수평 확장 방식 사용하기가 용이 • 기술 단일화, 관리 용이성 • 단순한 구조/간편한 코드 관리 • 통합 테스트에 용이	• Loosely Coupled • 서비스를 지향하여 각각의 모듈의 재사용성이 가능 • 기술·플랫폼 변화에 유연하게 대응 • Time to Market 신속히 대응하는 민첩성 향상 • 서비스의 인터페이스는 플랫폼에 독립적	• Low Coupling, High Cohesion • 복잡성의 문제 해결 • 서비스 단위 빠른 개발·배포 용이 • 서비스 단위 고효율 저비용 Scale-out 구조 • 서비스별로 독립적인 운영 • 유연한 개발과 운영
단점	• 특정 기능만 확장할 수 없음 • 부분 장애가 서비스 전체적인 장애 • 의존성이 높아 신규 개발 추가 및 변경의 어려움 • 배포 및 재기동 시간이 오래 걸림 • 코드 양 증가할수록 복잡도가 심각하게 증가 유지보수 어려움	• 실 성공사례 부족, 구축 비용 과다 • 성능 문제, 보안 문제 취약 • 서비스 도출, 방법론 부재 • 하나의 DB를 사용한다는 점에서 끊어질 수 없는 의존성이 존재 • 정책변경에 대한 민첩한 대응 미흡 • 일관성 부재	• 서비스가 분리되어 테스트와 트랜잭션의 복잡도 증가, 많은 자원 필요 • API 관리의 중요성 증가 • 테스트, 배포, 관리의 복잡성 • 데이터 transaction의 일관성 유지 어려움 • 통합 테스트가 어려움
특징	• 일반적으로 성능향상을 위해 Scale Up을 고려(서버 Spec 향상) • 소스 코드는 하나의 프로젝트로 구성, 단일 패키지로 배포 • 단일 코드 베이스 (하나의 프로젝트) - 결합도:매우 높음 - 확장성:매우 낮음 - 기술적 난이도:쉬움 - 비용:낮음	• 공통된 서비스를 ESB에 모아 공통 기능의 집합 통해 서비스 제공 • 서비스들을 느슨하게 연결, 각 컴포넌트를 독립적으로 운영 조립 가능 (하나 또는 여러 서비스) - 결합도:낮음 - 확장성:높음 - 기술적 난이도:어려움 - 비용:중간	• '작은 서비스'를 지향 • 각각의 서비스들은 API를 통해서만 데이터 교환 • 서비스들은 비즈니스 단위 구성 • 서비스들은 최소한의 중앙 집중식 구성 (독립된 여러 서비스) - 결합도:매우 낮음 - 확장성:매우 높음 - 기술적 난이도:어려움 - 비용:매우 높음

3. Ensemble Architecture 적용

3.1 Layer

(그림 2) Layer 별 아키텍처

실제 사용자가 최종 데이터를 UI(User Interface)로 확인하거나, 단순 데이터를 처리/전송하는 Client Layer 영역은 Monolithic이 더 적합하다. 본 시스템의 특성상 서비스를 하는 여러 국가와 지역에 따라 사용자 요구에 맞는, 서비스 지역별 디자인이 특화된 서비스를 하기 위해서는 Customizing이 필수인데, 그러한 요구사항은 서비스 이용자마다 불특정적이고 디자인 요구사항 또한 광범위하므로 Monolithic을 활용하여 빠르고 가볍게 설계하는 것이 더욱 효율적이다.

또한, 비즈니스를 처리하는 영역이 아닌 View 영역과 단순 데이터 처리/전송하는 영역이기에 시스템의 무거움을 고려하지 않고도 접근할 수 있다는 장점이 있다.

다음은, 비즈니스 로직을 처리하는 서비스 Layer 영역은 MSA가 더 효율적이다. MSA는 초기 설계 단계에 큰 비용이 소용될 수 있으나 본 시스템은 서비스 Layer, 즉 비즈니스를 처리하는 영역은 어느 지역에 서비스를 제공하건, 처리 기능들에 대한 틀은 대부분이 정해져 있다고 볼 수 있다.

시스템을 플랫폼화하여 비즈니스 처리 영역에 대한 기능들을 카테고리 묶음으로 분류하여 기본 Frame work를 만들고 화면 디자인이나 Out-put에 따라 ERD(Entity Relation Diagram)만 변환, 기능들은 세분화하여 시스템을 디자인하면 지역에 따라 특화된 기능만 제공/구성하고, 불필요 기능들은 제거하면서 보다 생산성 측면에서 좋은 시스템을 구현할 수 있다.

위치, 경로, 탑승, 예약 등에 대한 서비스 단위 기능들은 본 시스템을 사용하는 지역에 따라 기능 자체가 변화된다기보다, 그 안에 담기는 데이터들의 차이기 때문에 비즈니스를 처리하는 영역은 MSA로 구성하는 것이 가용성과 확장성에 있어서 유리하다.

마지막으로 외부통신 Layer 영역은 SOA가 적합하다. 사실 MSA가 SOA에서 진화된 형태여서 외부통신 Layer 또한 MSA를 적용할 수도 있다.

그러나 연계 대상이 특정되지 않고, 시스템을 제공하는 지역에 따라 통신대상 기관이 각각 다르며, 데이터의 포맷과 규격 또한 상대 기관마다 차이가 있어서 이용할 서비스에 따라 구성하는 SOA가 더욱 적합하다고 생각한다. 결제 서비스를 예를 들더라도 금융기관마다 표준화된 포맷을 이용한다기보다 기관이 API를 제공함으로써 대상 기관 기술요소에 맞춰 시스템과 서비스를 구성하는 것이 일반적이다. 때문에, MSA를 고려하는 것보다 서비스 지향 아키텍처를 적용하는 것이 시간과 비용면에서 조금 더 유리하다고 볼 수 있다.

3.2 Service

시스템에 제공하는 서비스 단위에 따라 각각의 아키텍처 방법론을 적용할 수 있다.

본 시스템에서 사용자에게 직접 마주치는 서비스는 크게 사용자용 Mobile, 관리자용 Web, 버스용 단말기로 나누어지는데, 이러한 단위별로 효율적인 아키텍처 방법론을 나누어서 적용할 수 있다.

(그림 3) Service 별 아키텍처

먼저 사용자용 Mobile 부분은 SOA 적용 부분과 MSA 적용 부분을 다시 나누어 볼 수 있는데, 위에 언급한 대로 MSA는 SOA가 진화된 형태이기 때문에 SOA로 구성된 전체를 MSA로 구성할 수도 있다.

하지만 앞서 예를 들어 설명한 것과 같이 결제 처리 또는 인증처리 등 외부통신을 처리하는 부분에서는 SOA를 적용하는 것이 더 합리적이다.

절에서 설명한 것과 같이 Monolithic으로 구현하는 것이 좋다고 언급했으나, 본 절에서는 서비스 단위 관점으로 아키텍처 방법론을 설명한다.

서비스 내의 다중 트랜잭션이 하나의 구성으로 묶어야 하는 부분은 SOA를 적용하는 것이 더욱 안정성이 있다. 이러한 서비스들은 MSA로 세분화 해봐야 결국 서비스 내에 종속되어 처리되기 때문에 MSA 설계에 대한 비용을 굳이 투여할 필요가 없다. 하지만 가입, 알림, 경로 탐색 등과 같이 독립적으로 분산처리가 가능한 영역은 MSA를 적용하기 좋은 형태이다. 일련의 처리 흐름에 있어서 독립적으로 구성이 가능한 서비스들은 세분화하여 구조를 설계하는 것이 향후 시스템을 제공하는 부분에 따라 가변적으로 운용할 수 있다는 장점이 있다.

관리자용 Web 서비스 경우 주로 활용되는 것이 시스템의 이동현황, 호출현황 등 현재 상태에 대한 모니터링 혹은, 데이터의 집계나 통계 바탕의 실시간 정보를 요구하는 서비스가 주로 구성되어있다. 이러한 서비스들은 데이터가 적재됨에 따라 그 즉시 활용할 수 있도록 Monolithic을 적용한 설계안이 간편하고 실용성이 있다. 데이터를 활용하여 있는 그대로 표출하거나 일부 가공을 통해 관리자에게 빠른 정보전달이 주인 관계로 서비스 지향이나 세분된 아키텍처 방법론 적용에 대한 비용을 줄이고 명시적으로 간단한 구성으로 빠른 설계와 개발을 할 수 있는 Monolithic이 더 효율적이다.

버스용 단말기에서 동작하는 서비스 경우 위치전송, 광고 서비스 등 시스템이 제공하는 국가나 지역 특성, 현지 서비스 기관의 다양성을 예상하기 힘듦에 따라 Monolithic을 적용하기에는 수용성과 확장성이 다소 부족하므로 SOA가 조금 더 적합하다. 마찬가지로 MSA를 적용하여 아키텍처를 구성할 수 있지만, Layer 절에서 언급한 외부통신 영역에 대한 특성이 같이 존재하여 SOA가 더욱 적합한 아키텍처라고 할 수 있다.

3.3 Function

(그림 4) Function 단위별 아키텍처

전체 시스템구조에서 기능(Function) 단위로 아키텍처 방법론을 적용할 수 있는 부분을 구분하여 표기하였다. 기능 관점으로 적합한 아키텍처 방법론을 생각해보면, Layer, Service 관점과는 또 다른 분류 형태를 볼 수 있다. 사용자가 직접 Action을 취하는 부분이나, Client 영역과 직접 맞닿아 있는 부분은 1절에서 언급한 것과 비슷한 논리로 Monolithic의 형태로 구성하여 시스템을 제공하는 서비스 지역 특성에 맞게 Customizing 설계 및 개발을 할 수 있도록 분류하였고, 기능상 외부통신과 연계된 부분은 2절에서 언급한 것과 같은 논리로 SOA 방법론으로 분류하였다.

본 절에서는 기존 Monolithic으로 제안한 통계 및 현황이 기능을 MSA 형태로 구성했을 때 또 다른 이점을 설명하고자 본 절에서는 통계 및 현황 부분을 MSA 카테고리로 분류하였다. 현황이나 통계 등은 기능상 세분화하여 별도의 분산처리로 설계안을 가져갈 수 있는데, 실시간성 데이터가 필요하지 않은 일 집계, 월 집계, 이력 성 로그 등은 MSA를 적용하여 구성하면 별도의 작업공간을 두어 시스템 전반적인 부하를 훨씬 줄일 수 있다. 이에 대한 아키텍처 구성 비용은 조금 더 발생하겠지만, 시스템 안정성 측면에서는 더 안정적으로 구성할 수 있다. 또한, 기능 관점으로 아키텍처를 고려하게 되면 개발할 때도 병렬로(Parallel) 기능 설계 및 개발이 진행될 수 있다는 장점이 있다.

4. 결론 및 향후 연구

아키텍처 방법론을 적용하기 위한 View 또는 경계에 따라 방법론에 대한 적용 범위를 달리할 수 있다.

이는 어떤 관점으로 시스템을 바라보느냐에 따라 그 관점에서 실용적인 방법론을 택하는 것은 시스템 설계 및 구축에 있어서 비용 절감 측면과 안정성, 효율성에 영향을 미친다.

본 논문에서는 MOD 시스템을 대상으로 하여 앙상블 아키텍처 방법론을 적용했을 때 발생하는 이점에 관해서 연구하였다. 앙상블 아키텍처를 적용한 설계 방법론은 단일 아키텍처 방법론을 적용했을 때보다 시스템의 유연성과 확장성을 가질 수 있다.

또한, 시스템의 플랫폼화에 있어서 향후 잠재 서비스 지역에 대한 설계 확장 방안이 될 수 있다. 지속해서 연구되고, 증명되고, 선행연구 들이 상호보완됨에 따라 그것들이 가지는 장점의 조합으로 또 하나의 방법론이 만들어질 수 있다.

제시한 앙상블 아키텍처 방법론에 대한 개념증명(Proof of Concept)을 수행하고, 향후 이론과 실천이 합치되는 설계 방법론을 넘어 개발 및 테스트를 고려한 확장 연구를 진행하고자 한다.

참고문헌

[1] 나태흠, 박평구, 류호용, “마이크로 서비스 아키텍처 기반 가상 인프라 매니저 설계 및 구현”, 디지털콘텐츠학회 논문지 제19권, 제4호4, 2018.

[2] 윤서연, 정일호, 이춘용, 김혜란, 육동형, 김광호, 이재현, “융합 빅데이터를 활용한 교통수요 추정 개선 연구”, 국토연구원, 2016.

[3] 박소현, 유혜영, “SOA와 WOA의 통합 아키텍처 설계에 관한 연구”, 정보처리학회지논문지D 제17-D권 제5호, 2010.

[4] 전철호, 서보민, 이근혁, 전현식, 박현주, “MSA가 적용된 서비스에서 생산되는 데이터의 효율적인 분석을 위한 연구”, 한국정보과학회 학술발표논문집, 114-1146, 2018.