Journal of the Korean Institute of Industrial Engineers
Vol. 47, No. 3, pp. 242-254, June 2021.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2021.47.3.242

<Original Research>

상호운용적 스마트 공장을 위한 PMML · PFA와 자산관리쉘 통합 방법

신승준[1] · 이주홍[2] · 박정민[3] · 엄주명[3†]

[1]한양대학교 산업융합학부/ [2]한양대학교 산업데이터엔지니어링학과/ [3]경희대학교 산업경영공학과

Integrating PMML and PFA with Asset Administration Shell for Interoperable Smart Factories

Seung-Jun Shin[1] · Ju-Hong Lee[2] · Joung-min Park[3] · Jumyung Um[3]

[1]Division of Interdisciplinary Industrial Studies, Hanyang University
[2]Department of Industrial Data Engineering, Hanyang University
[3]Department of Industrial & Management System Engineering, Kyunghee University

The present work proposes a method to integrate Predictive Model Markup Language (PMML) and Portable Format for Analytics (PFA) with Asset Administration Shell (AAS) for interoperable smart factories. This method enables the exchange and sharing of data analytics models across heterogeneous manufacturing assets through their deployment into AAS, which specifies a unified and standardized format to represent digital models, data and technical functionalities of physical assets. This article includes : the conceptual model and process, information structure and system architecture for creating and deploying PMML and PFA-based data analytics models into AAS. This article also includes a prototype implementation. In the prototype, a server creates regression and neural network-driven energy prediction models for a production machine, represents the models based on the PMML and PFA schema, generates and responds to an AAS instance with its submodels embedding such PMML and PFA models; meanwhile, a client requests and receives the AAS instance on a web environment.

Keywords: Smart Factory, Manufacturing Intelligence, Interoperability, Asset Administration Shell, Portable Format for Analytics, Predictive Model Markup Language

1. 서 론

스마트 공장은 독일, 미국, 한국 등 제조강국들의 제조경쟁력 강화를 위한 범국가적 실행전략으로서, 현재는 개념에서 벗어나 실체성을 가진 구현 결과물로 가시화되고 있다(Moghaddam *et al*., 2008; Park *et al*., 2020). 이에, 각 국가에서는 현실을 감안한 스마트 공장 개발 및 전개를 위한 기준과 참고가 되는 참조 모델을 개발 중에 있다(Park *et al*., 2020). 그 중에서 가장 널리 알려진 참조 모델은 Reference Architecture Model Industrie 4.0(RAMI 4.0)일 것이다. RAMI4.0은 독일의 Platform Industrie 4.0 전략 구현을 위하여 생애주기, 가치사슬, 제조 구조 및 기능 계층의 필요 기술 및 표준들을 정의하는 참조 모델이다(Adolphs *et al*., 2015). RAMI4.0에서는 생애주기 · 가치사슬 측면의 수평적 통합, 제조 구조 측면의 수직적 통합을 제시하고 있다. 특히, 수평적 통합과 수직적 통합에 필요한 호환성을 제공하는 상호운용성(interoperability) 통합 모델을 제시하고 있다(Adolphs *et al*., 2015).

이 논문은 2018년 교육부와 한국연구재단 이공학개인기초연구지원사업의 지원을 받아 수행되었음(NRF-2018R1D1A1B07047100). 한양대학교(서울) 링크플러스 사업단 스마트팩토리(Varient4) 장비의 지원을 받아 수행된 연구 결과임[과제명 : LINC+ 사회맞춤형 산학협력 선도대학 육성사업].
† 연락저자 : 엄주명 조교수, 17104 경기도 용인시 기흥구 덕영대로 1732 경희대학교 산업경영공학과, Tel : 031-201-3695, Fax : 031-203-4004, E-mail : jayum@khu.ac.kr
2021년 2월 4일 접수; 2021년 4월 13일 수정본 접수; 2021년 4월 14일 게재 확정.

RAMI4.0에서 상호운용성의 중추적 역할로 정의된 것이 I4.0 컴포넌트(component)라고 명명된 제조 자산과 이에 대응되는 가상 객체 표현 모델이다(Adolphs *et al*., 2015). I4.0 컴포넌트는 각 제조 자산(예 : 설비, 장치, 부품, 제품, 소프트웨어, 휴먼 인터페이스 및 사람 등)을 컴포넌트화하고 이들 간의 수평적 · 수직적 구조화를 가능하게 한다. 나아가, 제조 자산들에 대한 식별, 종류, 통신, 데이터 및 기술적 기능성들에 대한 정보를 통일되고 규격화된 형태로 공유를 가능하게 한다. 즉, I4.0 컴포넌트는 일관적인, 규격화된 그리고 디지털적인 정보 컨테이너 안에 제조 자산의 정보들을 담게 되는데, 이 역할을 수행하는 것이 관리쉘(Administration Shell : AS)이며, 자산 객체가 투영된 것이 자산관리쉘(Asset Administration Shell : AAS)이다.

자산관리쉘은 I4.0 컴포넌트의 가상적이고 디지털적이며 능동적인 표현 모델로 정의되며, 제조 자산의 관리 인터페이스 계층 및 서브모델 계층으로 구성된다(Wagner *et al*., 2017). 예를 들어, 물리적 생산 설비는 가상의 자산관리쉘로 표현되며, 그 설비와 관련한 설계도면, 시뮬레이션 모델, 기능 블럭, 매뉴얼, 센싱 데이터 등 데이터와 기술적 기능성(technical functionality)을 포함한 서브모델들을 보유하게 된다. 그리고 관리 인터페이스를 통하여 해당 자산의 객체 식별, 통신 및 데이터 교환을 가능하게 한다. 자산관리쉘의 첫 번째 규격은 2018년, 두 번째 규격은 2019년에 공개되었다(Federal ministry for economic affairs and energy, 2019).

한편, 스마트 공장의 핵심 목표 중 하나는 제조 지능화(manufacturing intelligence)이다. 제조 지능화는 제조 자산의 데이터로부터 통찰력과 예지력을 획득하고, 이를 제조 자산의 자동화, 자율화 및 협업화에 활용함으로써, 생산성, 친환경성, 유연성 및 실시간성을 향상시키고자 하는 것이다(Davis *et al*., 2012). 데이터로부터의 통찰력과 예지력은 데이터 애널리틱스(data analytics)를 통한 설명적(descriptive), 진단적(diagnostic), 예측적(predictive) 및 처방적(prescriptive) 모델의 생성과 활용에 의하여 획득될 수 있다(Belhadi *et al*., 2019). 이러한 모델들이 제조 자산에서 적용되고 타 자산들과 교환되고 공유됨으로써 목표 성능 향상을 위한 자동화, 자율화 및 협업화를 가능하게 하는 것이다. 최근의 제조 지능화는 상호운용성 보장을 강조하고 있다(Ministry of economy and finances in france, 2018). 왜냐하면 제조 지능화의 대상이 제조 자산별 국소적 영역에서 벗어나 모든 제조 자산을 망라하는 시스템적이고 광역적인 영역으로 확대되고 있기 때문이다. 따라서, 이기종 기기 및 시스템들이 아무런 제약 없이 데이터가 서로 호환되고 공유되는 환경을 구축함으로써 개별 제조 자산뿐만 아니라 전체 시스템을 아우르는 상호운용적 제조 지능화의 실현이 필요하다.

제조 분야에서는 수직 · 수평적 데이터 통합, 가치사슬 통합, 컴퓨터지원 도구 통합, 제품수명주기 통합 그리고 클라우드 환경 통합 등 다양한 관점의 상호운용성이 존재한다(Shin and Meilanitasari, 2020). 추가적으로, 모델 관점의 상호운용성이 필요하다. 모델 상호운용성은 제조 자산들이 언제 어디서나 데이터 애널리틱스 모델들을 교환 및 공유하며 이를 활용하는 환경을 의미한다. 이 모델 상호운용성을 통하여 다양한 제조 지능화 시나리오 개발이 가능하다. 예를 들어, 생산설비가 목표 성능에 대하여 기계학습 기반 예측적 모델을 생성하고 모델 저장소로 전송한 후, 필요할 때마다 다시 호출하여 그 모델을 활용할 수 있다. 또한, 생산라인의 목표 성능 예측이 필요할 때, 제조실행 시스템이 다수의 생산설비들로부터 수집된 예측적 모델들을 별도의 변환 작업 없이 활용하여 즉각적인 예측을 가능하게 한다. 그리고 한 생산설비에서의 예측 모델을 다른 생산설비에서 활용할 때, 두 생산설비 간의 막힘없는 모델 교환을 가능하게 한다.

모델 상호운용성을 위해서는 데이터 애널리틱스 모델의 표현이 선행되어야 한다. 이는 데이터 애널리틱스 모델이 실제적이고 정형적인 형태로 표현된 후에야 모델 교환과 공유가 가능해지기 때문이다. 대표적인 데이터 애널리틱스 모델의 표현 언어로는 Predictive Model Markup Language(PMML)와 Portable Format for Analytics(PFA)가 있다. 두 언어는 Data Mining Group (DMG)에서 개발한 것으로서, 데이터 분석 분야에서는 사실상 표준으로 활용되고 있다. PMML은 통계 및 예측적 모형뿐만 아니라 데이터 전 · 후처리를 표현하는 스키마로 구조화된 XML 기반 언어이다(Guazzelli *et al*., 2009). 반면, PFA는 PMML로부터 파생된 것으로서, PMML의 컨텐츠를 프로그래밍 가능하고 확장적이며 경량적으로 표현한 JavaScript Object Notation(JSON) 기반 언어이다(Pivarski *et al*., 2016). 그러나 PMML과 PFA는 데이터 분석 영역의 언어이므로 제조 분야에 특화된 것은 아니다. 제조분야에서 이 언어들을 활용하려면, 수작업으로 PMML과 PFA 지원 데이터 분석 도구(예 : SPSS, KNIME, R)를 사용해야만 한다. 아니면, 파일 형태로 교환되는 PMML과 PFA 문서를 처리하는 인코더와 디코더를 개발해야 한다. 그러나 두 경우 모두, 제조 자산들이 자율적이고 협업적인 방식으로 데이터 애널리틱스 모델을 교환 및 공유하기에는 한계가 있다. 왜냐하면 별도의 구현을 통하여 제조 자산 정보와 연계해야 하며, 다양한 기기와 시스템 간 호환을 위해서 개별적인 인코더와 디코더의 개발이 필요하기 때문이다.

본 논문에서는 상호운용적 제조 지능화 구현을 위하여 제조 자산이 생성한 PMML과 PFA 기반 데이터 애널리틱스 모델을 자산관리쉘에 탑재하여 교환 및 공유하기 위한 방법을 개발한다. 구체적으로, 자산관리쉘로 대변되는 I4.0 컴포넌트가 데이터 애널리틱스를 통하여 PMML과 PFA 형태의 예측적 애널리틱스 모델을 생성하고, 이러한 예측 모델이 자산관리쉘이라는 정보 컨테이너의 서브모델로 탑재한 후, 이 자산관리쉘 정보를 다른 I4.0 컴포넌트와 교환하고 공유하는 방법이다. 이 방법은 데이터 표준화를 통한 상호운용성을 증진시키기 위함이다. 제조 지능화를 위해서는 제조 자산의 가상 객체들이 자신의 데이터 애널리틱스 모델 또한 객체 형태로 내재화하고, 이들을 예측 및 최적화 용도로 활용하는 환경의 구축이 필요하다. 이를 위하여, 제조 자산 표현 모델의 표준 중 하나인 자산관리

쉘과 데이터 애널리틱스 모델의 표현 언어의 표준인 PMML 및 PFA를 통합하는 것이다. 본 논문에서는 자산관리쉘과 데이터 애널리틱스 통합을 위한 개념 모델, 과정, 정보 구조, 시스템 기능 구조의 설계 그리고 시제품 구현을 설명한다. 시제품은 실험용 열처리 공정 설비를 대상으로 자산관리쉘의 서버-클라이언트를 구현하였다. 서버 측에서는 선형회귀 및 인공신경망 기반 에너지 예측 모델 생성, PMML 및 PFA 규격으로의 변환, 자산관리쉘의 생성과 서브모델 등록이 이루어진다. 클라이언트 측에서 자산관리쉘 인스턴스를 요청(request)하면, 서버 측에서는 해당 자산관리쉘 인스턴스를 반환(response)하게 된다.

본 논문의 구성은 다음과 같다. 제 2장에서는 자산관리쉘과 PMML · PFA의 관련연구를 소개한다. 제 3장에서는 자산관리쉘과 데이터 애널리틱스 통합의 개념 모델, 과정, 정보 구조 및 기능 구조 설계를 설명한다. 제 4장에서는 구현을 서술하며, 제 5장에서는 결론을 맺는다.

2. 관련연구

2.1 자산관리쉘 (Asset Administration Shell)

자산(asset)이라고 함은 조직에서 가치가 있고 소유권을 가진 물리적 또는 논리적 객체(object)를 의미한다(Ministry of Economy and Finances in France, 2018). 자산에는 기계, 컴퓨터, 액츄에이터, 센서, 부품, 제품 및 사람과 같은 물질적 자산과 소프트웨어, 문서, 정보 및 서비스와 같은 비물질적 자산이 있다. 그러므로 제조 시스템에 존재하는 모든 구성요소들을 제조 자산이라고 정의할 수 있다. 그러나 제조 자산들은 매우 다양한 종류, 용도, 공급자와 사용자들이 존재하며, 자산들 간 데이터 교환 방법 또한 매우 다양하다. 이로 인하여 자산 간의 정보 고립 및 단절 문제가 지속적으로 발생하였다. 이러한 문제들을 해결하려면, 자산들의 객체를 식별하고 정보를 공통적으로 표현함과 동시에, 자산들 간 관계 및 계층을 유연하게 구성하며, 자산들 간 정보 교환을 일관적으로 수행하는 상호운용성 획득이 필요하다. 즉, 표준화 접근법이 필요하다. 이러한 목적에 기인하여 자산관리쉘이 개발되었다.

<Figure 1>은 자산관리쉘의 정보 구조이다. 자산관리쉘은 헤더(header)와 바디(body)로 구성된다. 헤더는 물리적 자산의 객체식별(identification asset)과 해당 자산관리쉘의 객체식별(identification administration shell)을 포함하여, 관계 및 보안 등의 메타데이터를 담는다(Ministry of Economy and Finances in France, 2018). 식별자(identifiers)의 경우, 글로벌하게 고유하도록 International Registration Data Identifier(IRDI), International Resource Identifier(IRI), Uniform Resource Identifier(URI) 등을 사용한다. 커스터마이즈된 식별자의 활용도 가능하다. 한편, 바디는 서브모델들의 집합으로 구성되며, 서브모델은 데이터(data)와 기능(function)을 표현한 것이다(Ministry of Economy and Finances in France, 2018). 즉, 자산관리쉘의 컨텐츠 영역이다. 데이터는 자산에 대한 디지털 데이터를 의미하며, 설계도면, 시뮬레이션 모델, 기능 블럭, 매뉴얼, 센싱 데이터 등을 담을 수 있다. 기능은 자산의 기술적 기능성(technical functionality)을 의미하며, 기술적 기능성을 Application Programming Interface (API)로 노출하는 형태로 구현된다(Ministry of Economy and Finances in France, 2018). 기능의 예로는 자산의 수행 공정 종류, 공정 역량과 성능, 상태 모니터링, 에너지 효율 등을 포함하며, 자산으로부터의 부가 가치 창출을 위한 서비스들을 추가할 수 있다. 또한, <Figure 1>에서는 표현이 배제되어 있으나, 데이터와 기능 간의 상호 참조 및 활용도 가능하다.

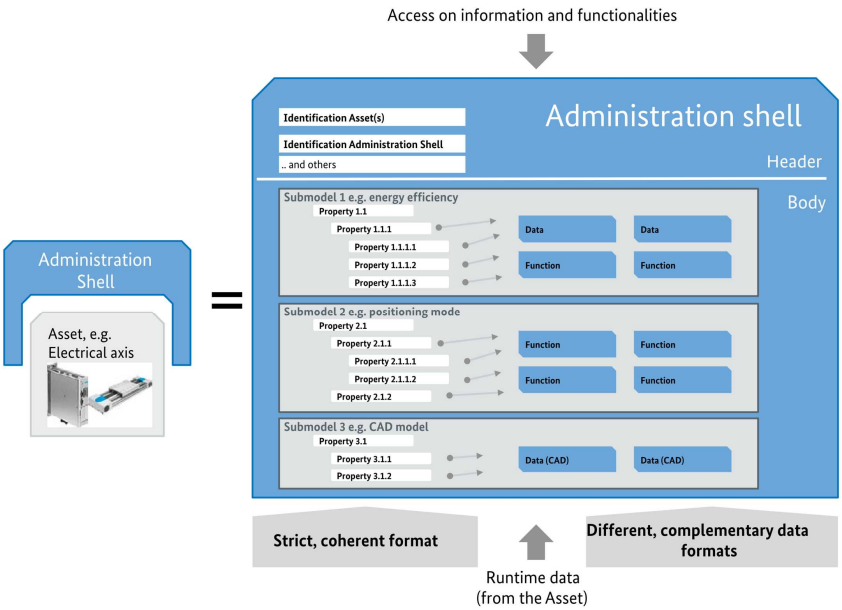


Figure 1. General Structure of Asset Administration Shell(Ministry of Economy and Finances in France, 2018)

각 서브모델은 규격화된 형식뿐만 아니라 비규격적인 다양한 형식으로의 구조체를 구성할 수 있다. 규격화된 구조체의 예로는 International Electrotechnical Commission(IEC) 61360이 있다. 이 표준은 제품 사전(product dictionary)을 위한 정보 모델로서, 제품들을 클래스로 분류하고 각 제품별 속성 정보들을 규격화한 것이다(Ministry of economy and finances in france, 2018). 또한, 각 서브모델은 다양한 데이터 요소(element)들을 담을 수 있다. 데이터 요소의 예로는 프로퍼티(특정값), 프로퍼티 범위, 파일, Binary Large Objects(BLOB), 오퍼레이션, 이벤트, 엔티티 등이 있다(Federal Ministry for Economic Affairs and Energy, 2019). 그리고 자산관리쉘은 XML, JSON, Resource Description Framework(RDF) 또는 Open Platform Communications Unified Architecture(OPC UA) 언어로의 표현이 가능하다(Federal Ministry for Economic Affairs and Energy, 2019). 이러한 포괄적이면서 유연한 구조를 통하여 자산에 대한 모든 정보의 표현 및 공유가 가능해진다.

자산관리쉘의 종류는 타입(type)과 인스턴스(instance)가 있다(Federal Ministry for Economic Affairs and Energy, 2019). 자산관리쉘의 타입을 정의한 후, 그 타입 정의를 가져와서 인스턴스를 생성하는 것이다. 각각 객체지향 프로그래밍에서의 클래스와 인스턴스 개념과 유사하다. 일반적으로, 제조 자산의 개발 단계에서 자산관리쉘 타입을 정의한 후, 제작 또는 사용 단계에서 그 타입을 참조하여 자산관리쉘 인스턴스를 생성하는 것이다.

자산관리쉘 관련 연구는 기술 태동기 상에 있으며, 요구사항과 개념 정립 그리고 규격 개발 위주로 진행 중이다. Grangel-González *et al*.(2016)은 RDF 및 Web Ontology Language를 이용한 시맨틱기반 자산관리쉘 표현방법을 개발하였다. Tanktik and Anderl(2017)은 자산관리쉘을 Object Memory Model로 구성 및 표현함으로써, 인터넷 상의 자산관리쉘 상호운용성을 보여주었다. Marcon *et al*.(2018)은 작업자용 휴먼-머신 인터페이스의 자산관리쉘에 대한 시제품을 개발하였다. Cavalieri *et al*.(2019)과 Fuchs *et al*.(2019)은 자산관리쉘의 OPC UA 통합 정보모델을 개발하였다. 참고로, OPC UA는 이기종 시스템 및 기기 간 규격화된 데이터 교환을 가능하게 하는 상호운용성 인터페이스로서(Cavalieri *et al*., 2019), RAMI4.0에서는 자산관리쉘과 함께 OPC UA를 상호운용성을 위한 기술로 추천하고 있다. Trunzer *et al*.(2019)은 Industrie 4.0을 위한 범용적 시스템 아키텍처를 제안하였다. 이 아키텍처는 데이터 관리 및 통합 버스를 중심으로 기업 간 데이터 공유, 서비스 오케스트레이션 및 실시간 역량을 도모하고 있다. Luder *et al*.(2020)은 AutomationML로 표현되는 엔지니어링 데이터의 자산관리쉘 탑재를 위한 정보 구조 및 방법을 개발하였다.

국내의 경우는 일부 선도 기관에서 연구를 진행 중이나, 아직은 미진한 상태이다. Ye and Hong(2019)은 자산관리쉘의 구조 및 기술적 요구사항 등의 필수정보를 담을 수 있는 서브모델 템플릿을 개발하였다. Park *et al*.(2019) 및 Park *et al*.(2020)은 염색가공 산업의 에너지 효율화를 위하여 자산관리쉘의 특성을 반영한 디지털트윈 어플리케이션 및 가상표현 모델을 개발하였다.

향후에는 자산관리쉘을 활용한 정보 컨텐츠 확장과 이를 이용한 시나리오 및 응용 기술 개발로 전개될 것으로 예상한다. 이는 자산관리쉘 규격 및 구현용 지원도구들(예 : 소프트웨어 개발 키트, 탐색기)이 공개되기 시작했기 때문이다. 기존에는 자산관리쉘 구현의 모든 작업을 개별적으로 수행해야 했다면, 이제는 지원도구를 이용하여 시나리오의 구현 및 기술 개발로의 집중이 가능해 질 것이다.

2.2 PMML과 PFA

PMML은 데이터 전・후처리 및 데이터 애널리틱스 모델을 표현하기 위한 XML기반 개방형 표준 언어이다(Guazzelli *et al*., 2009). 현재, 18개의 데이터 애널리틱스 모델(회귀분석, 인공신경망, k-최근접 이웃 등)이 규격화되어 있다(버전 4.4 기준)(Data mining group, 2019). <Figure 2>는 PMML 모델의 정보 구조이다. 헤더(header)는 PMML 문서의 메타데이터, 데이터 사전(data dictionary)은 데이터 필드의 변수 명칭 및 타입 선언, 데이터 변환(data transformation)은 데이터 전처리, 모델(model)은 상세적 모델을 표현한다. 그리고 모델 하부의 마이닝 스키마(mining schema)는 데이터 필드의 상세 정보, 타겟(target)은 출력 데이터 필드 선언 및 후처리, 모델 세부사항(model specifics)에서는 각 모델의 종류, 구조 및 변수들을 상세화한다.

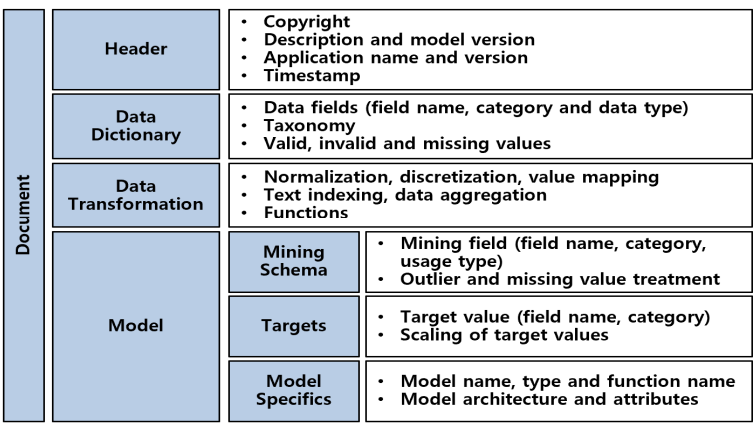


Figure 2. Structure of PMML(re-edited from(Guazzelli *et al*., 2009))

```
{"input":
  {"type": "record",
   "name": "Input",
   "fields": [
      {"name": "datum", "type": {"items": "double", "type": "array"}},
      {"name": "model", "type": {"fields": [{"type": {"items": "double", "type": "array"}, "name": "coeff"},
                                            {"type": "double", "name": "const"}], "type": "record", "name": "Record"}} ]},
 "output":
    "double",
 "action":
    {"model.reg.linear": [
        "input.datum",
        "input.model" ]} },
```

Figure 3. Example of PFA Document(Data Mining Group, 2015)

한편, PFA는 데이터 애널리틱스 모델을 프로그래밍 가능하고 확장적이고 경량화한 JSON 기반 언어이다(Pivarski *et al.*, 2016). PMML은 정적 형태의 표현에 국한되다 보니, 확장적이지 못하고 데이터 처리 플랫폼에서 동적인 작동이 불가능하다. 반면, PFA는 개발자가 프로그래밍하여 분류기, 예측기, 평활기, 필터 등을 만들 수 있고, Python과 Scala 등 다른 플랫폼과의 호환이 가능하다. <Figure 3>은 선형회귀 모델에 대한 PFA 문서 예시다. 입력(input)은 입력 데이터 명칭 및 타입을 정의하고, 출력(output)은 출력 데이터 타입을 정의한다. 액션(action)에서 수행할 함수를 정의하는데, 선형회귀의 경우는 model.reg.linear로 정의한다(Data mining group, 2015).

제조 분야에서 PMML은 데이터 애널리틱스 모델의 표현, 확장 및 검증의 용도로 주로 활용되었다. 모델 표현은 모델의 교환 및 공유를 위하여 PMML을 표현 언어로 사용한 것이다. O'Donovan *et al.*(2016)과 Lechevalier *et al.*(2018)은 각각 서포트 벡터 머신과 인공신경망 모델을 PMML로 표현하여 생산설비의 예측 모델로 활용하였다. 모델 확장은 모델의 사용성과 추적성을 향상시키기 위하여 기존 PMML 스키마에 추가적인 정보를 합성한 것이다. Lechevalier *et al.*(2015)은 인공신경망 모델의 데이터베이스 저장 및 호출을 위하여 구조적인 메타 모델을 기존 PMML 구조에 추가하였다. Zhang *et al.*(2019)은 제품, 공정계획 및 설비 정보를 기존 PMML기반 모델과 연결하는 정보 모델을 개발하였다. 모델 검증은 신규 PMML 모델에 대한 검증을 제조 분야에서 수행한 것이다. Park *et al.*(2017)은 가우시안 프로세스 회귀 모델의 신규 PMML 스키마를 절삭공정의 에너지 예측 시나리오에서 검증하였다. Nannapaneni *et al.*(2018)은 베이시안 네트워크 모델의 PMML 스키마를 용접공정의 에너지 예측 시나리오에서 검증하였다. 반면, PFA는 제조 분야에서의 활용가능성을 언급한 수준이며(Lechevalier *et al.*, 2018; Zhang *et al.*, 2019), 활용 사례 및 구현은 거의 없다. Lechevalier *et al.*(2015)은 절삭가공의 에너지 예측을 위한 인공신경망 모델을 PFA 문서로 생성한 사례가 있다.

3. 방 법

본 장에서는 PMML 및 PFA 기반 데이터 애널리틱스 모델과 자산관리쉘 통합을 위한 방법을 소개한다. 여기서, 방법은 개념 모델, 과정, 정보 구조 및 시스템 구조로 구성되며, 각 절에서는 이들을 설명한다. 쉬운 이해를 돕고자, 대표적 제조 자산인 생산설비를 중심으로 설명한다.

3.1 개념 모델

상호운용적 제조 지능화를 위한 데이터 애널리틱스 모델과 자산관리쉘의 통합을 위한 접근법의 정의가 필요하다. <Figure 4>는 이 접근법을 개념적으로 도식화 한 것이다. 일반적으로, 자산의 수명주기는 요구사항 분석(requirement), 설계(design), 제작(implementation), 운영 및 유지보수(operation/maintenance) 그리고 폐기(decommission)의 단계를 거치게 된다(Ministry of economy and finances in france, 2018). 2.1절에서 설명한대로, 자산관리쉘의 인스턴스는 제작 및 운영 단계에서 생성된다. 이 중에서, 운영 단계는 자산의 사용 영역에 해당한다. 자산의 사용자는 자산의 가용성과 생산성을 증진시키고자 제조 지능화를

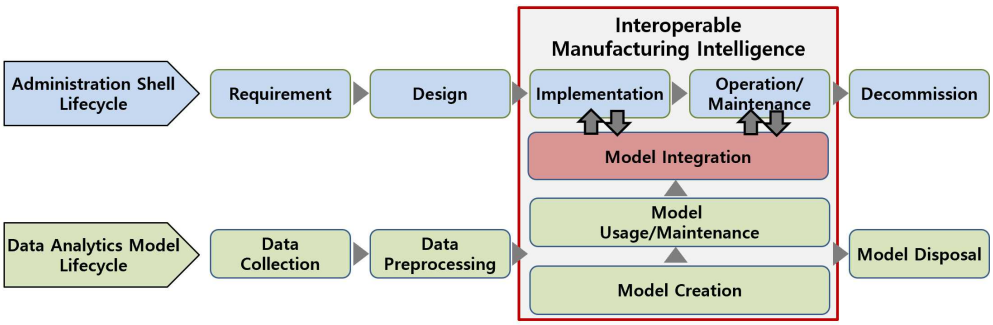


Figure 4. Conceptual Model

위한 활동을 수행할 것이다. 이러한 활동의 결과는 데이터 애널리틱스 모델로 구체화 된다. 그렇다면, 자산의 사용자는 데이터 애널리틱스 모델들을 어떻게 정형화하고, 어떻게 체계적으로 관리, 사용 및 교환하는 것이 효율적이고 효과적인지에 대한 기술적 문제를 고민할 것이다. 가장 쉬운 해결책은 데이터 애널리틱스 모델의 대상이 되는 그 자산을 중심으로 모델을 생성, 보유 및 활용하는 것이다. 즉, 해당 자산에 대한 자산관리쉘이라는 정보 컨테이너에 데이터 애널리틱스 모델을 담아서 사용하는 것이다. 이는 자산관리쉘의 목적이기도 하며, 자산관리쉘이 객체식별, 관계, 보안 기능과 함께 데이터와 기술적 기능성의 저장 기능이 있으므로 구현 또한 가능하다. 즉, 자산관리쉘을 중심으로 데이터 애널리틱스 모델을 내재화하기 위한 모델 통합(model integration)이 필요하다.

그 다음은 상호운용성을 위하여 어떤 데이터 애널리틱스의 표현 모델을 선택하는지에 대한 문제가 있다. 쉽게 말하면, 데이터 애널리틱스 모델은 $y = F(X)+\varepsilon$ 함수 형태로 표현된다. 이러한 함수의 표현이 가능하다면 어떠한 언어(예 : R, KNIME, Scala, Phyton)라도 자산관리쉘과의 통합은 가능하다. 본 논문에서는 PMML과 PFA를 선택하였는데, 그 이유는 상호운용적 제조 지능화를 추구하고 있기 때문이다. PMML과 PFA는 데이터 애널리틱스 모델의 계층적 · 구조적 표현이 가능하고, XML과 JSON이라는 범용적이고 기계가 해석가능한 언어를 사용하며, 데이터 분석 영역에서는 사실상 표준 중 하나로 활용하고 있기 때문이다.

3.2 과정

<Figure 5>는 PMML · PFA 기반 데이터 애널리틱스 모델과 자산관리쉘의 통합 과정을 나타낸다. 예측적 애널리틱스의 경우, 자산으로부터 데이터 인터페이스를 통하여 데이터를 수집하게 된다. 그리고 데이터 애널리틱스 도구를 이용하여 수집된 데이터로부터 예측 모델을 생성하게 된다. 예측 모델은 다양한 기계학습 기법을 이용하여 목표 성능에 대한 수학적 함수로 도출된다. 이러한 함수 형태의 형식화 및 구조화를 위하여, PMML 스키마 및 PFA 스키마에 근간한 예측 모델의 래핑(wrapping)을 수행한다. 모델 래핑은 그 예측 모델을 XML 기반 PMML 형태의 문서 및 JSON 기반 PFA 형태의 문서로 인코딩하는 과정이다. 이와 동시에, 자산에 대응하는 자산관리쉘 객체식별자와 관련 인스턴스를 생성한다. 그 후, 자산관리쉘은 PMML · PFA의 문서들을 서브모델로 탑재한다. 이러한 과정을 통하여 자산관리쉘은 PMML · PFA 예측 모델들을 서브모델의 인스턴스로 내재화하게 된다. 이 때, 예측 모델들을 언어 형태(XML 또는 JSON), 존재 형태(파일 또는 인스턴스) 및 기계학습 기법 종류별로 다양하게 생성하는 것이 유리한데, 이는 모델 사용 시점에서 용도, 사용 환경, 선호도 등에 따라 모델을 선택적으로 사용하게 해준다.

상기 방법은 데이터 애널리틱스 모델을 표현한 PMML · PFA 인스턴스 또는 파일을 서브모델의 데이터로 저장한 후, 이 데이터의 호출은 API로의 접근을 통한 기능 형태로 이루어진다. 즉, 데이터와 기능이 상호 참조되는 것이다(2.1절 참고). 이는 데이터 애널리틱스 모델의 종류마다 모델 구조, 입출력 변수 및 매개변수들이 상이한데, 이러한 다양성을 구조화하고 표현한 기존의 표준을 활용함으로써 범용성을 높이기 위함이다. 만약, 데이터 애널리틱스 모델들이 서브모델의 기능으로만 존재한다면, 데이터 애널리틱스 모델이 API 내부에 은닉된 형태로 프로그래밍 되는 즉, 블랙박스화된 API 방식을 취하게 된다. 이 방식은 표준적인 정보모델로서의 역할과 데이터 애널리틱스 모델의 다양성에 대한 접근이 어렵다. 본 논문의 방법은 자체 정보모델의 개발을 최대한 배제하고, 기존 표준들을 최대한 활용하여 상호운용성을 향상시키고자 설계된 것이다. 참고로, Programmable Logic Controller(PLC)의 프로그램 언어를 표준화한 IEC 61131-3과 자산관리쉘 간의 통합 사례 또한 PLC 프로그램은 데이터로, 데이터 호출은 API의 기능으로 구현하였다(Cavalieri and Salafia, 2020).

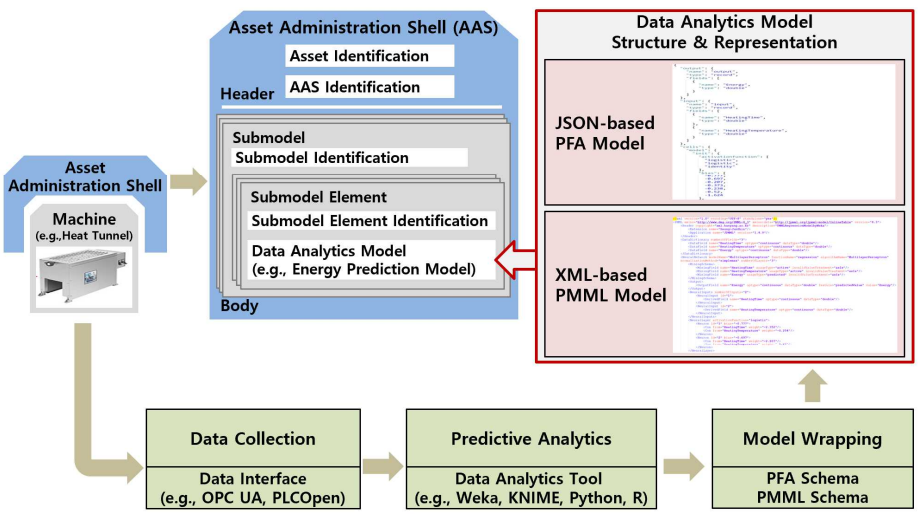


Figure 5. Conceptual Process

3.3 정보 구조

<Figure 6>은 PMML · PFA 모델 통합을 위한 자산관리쉘의 상세 정보 구조로서, Unified Modeling Language의 클래스 다이어그램으로 표현한 것이다. 기본적인 정보 구조는 자산관리쉘 규격의 것이다(Federal Ministry for Economic Affairs and Energy, 2019). 각 클래스에 대한 설명은 아래와 같다. 여기서, 클래스 상단에 관리 정보가 있는데, Identifiable(객체식별 번호 및 버전), HasSemantics(시맨틱 정의), Qualifiable(적격여부 확인), HasDataSpecification(데이터의 추가 속성 정의), HasKind (타입 또는 인스턴스 여부) 사항을 별도의 범용 클래스를 이용하여 정의하는 것을 의미한다.

- AssetAdministrationShell(자산관리쉘) : 자산관리쉘에 해당하는 최상위 클래스임. 고유한 객체식별자가 있어야 하며, 단수 개의 Asset을 가지며, 복수 개의 Submodel을 가질 수 있음.
- Asset(자산) : 자산관리쉘에 의해 표현되는 자산의 메타 데이터에 대한 클래스임. 여기서, 타입인지 인스턴스인지 선택함.
- Submodel(서브모델)과 SubmodelElement(서브모델 요소) : Submodel은 자산관리쉘의 디지털 데이터 및 기술적 기능성을 표현하는 클래스임. SubmodelElement는 추상형 클래스로서, Submodel과 합성의 관계이며 Submodel을 구성하는 여러 요소들을 포함함. DataElement 외에 다른 데이터 형태(오퍼레이션, 이벤트, 엔티티 등)를 가질 수 있음.
- DataElement(데이터 요소) : SubmodelElement를 상속받는 추상형 클래스임. 속성, 범위, 파일, BLOB 등의 데이터 형태를 서브 클래스로 두고 있음.
- Property(프로퍼티) : 범용적인 데이터 형태를 취하는 단수 개의 값을 갖는 클래스임. 이 클래스를 이용하면, PMML · PFA 모델의 클래스 인스턴스를 문자(string) 형태의 Submodel로 인스턴싱할 수 있음.
- File(파일) : 데이터 파일을 사용하기 위하여 파일 경로를 갖는 클래스임. 만약, PMML · PFA 모델이 파일형태로 저장되어 있다면, 해당 경로를 표현함으로써 Submodel로 가져올 수 있음.

3.4 시스템 구조

3.1절의 개념 모델을 구현하려면, PMML · PFA 모델을 통합한 자산관리쉘의 생성과 공유를 수행하는 시스템 측면의 기능 구조 정의가 필요하다. <Figure 7>은 시스템 구조를 나타낸다. 기본적으로 서버-클라이언트 구조에 근간한다. 자산관리쉘 클라이언트(asset administration shell client)가 자산관리쉘 정보를 요청하면, 자산관리쉘 서버(asset administration shell server)가 자산관리쉘 정보를 반환한 후, 클라이언트가 이 정보를 목적에 맞게 활용하는 흐름이다.

서버는 자산 정의 및 데이터 수집을 위하여 제조 현장의 물리적 자산 또는 레거시 시스템과의 연결이 필요하다. 물리적 자산과의 직접적 연결은 OPC UA 등의 필드레벨 데이터 인터페이스를 통하여 데이터 수집이 가능하다. 필드레벨 데이터가 제조 어플리케이션, 데이터베이스 등에 저장되는 경우, 이러한 레거시 시스템과의 간접적 연결이 가능하다. 서버는 모델 관리(analytics model management), 자산관리쉘 관리(administration shell management) 및 서버 등록(server registry) 모듈로 구성된다. 모델 관리는 데이터 애널리틱스 및 PMML · PFA 모델의 래핑을 수행한다. 자산관리쉘 관리는 자산관리쉘을 생성하고, PMML · PFA 모델을 서브모델로 등록하여 자산관리쉘 정보를 완성한다. 서버 등록은 이러한 자산관리쉘 정보에 대응하는 고유 식별자(예 : IRI, URI)를 생성한 후, 클라이언트로부터 요청이 발생하면 자산관리쉘 인스턴스를 반환하게 된다. 필요시, 자산관리쉘 정보를 데이터베이스에 저장한다. 한편, 클라이언트에서는 인증절차를 거쳐 서버에 접속하게 된다. 그리고 해당 고유 식별자를 이용하여 자산관리쉘 정보를 요청한다. 그 후, 서버로부터 반환된 자산관리쉘 정보 및 PMML · PFA 모델을 해석하여 목적에 맞게 사용한다.

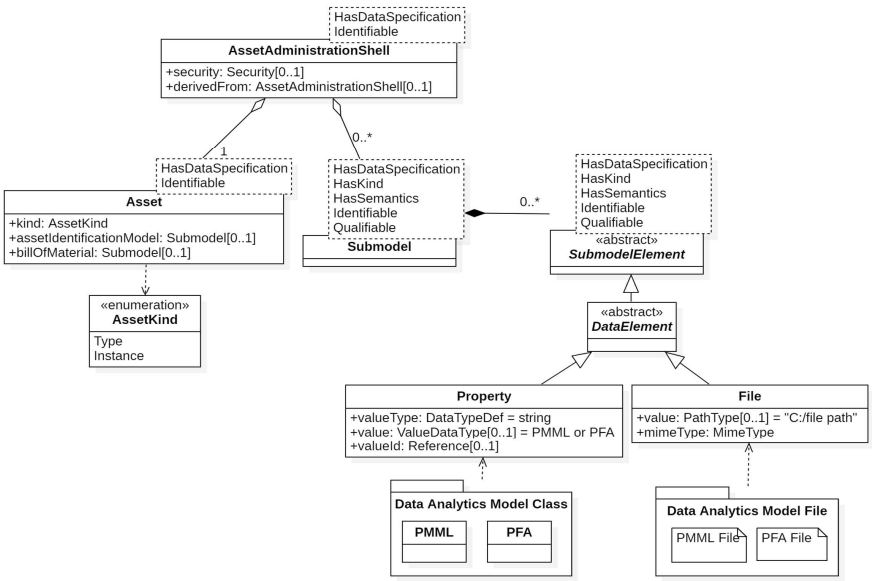


Figure 6. Information Structure.

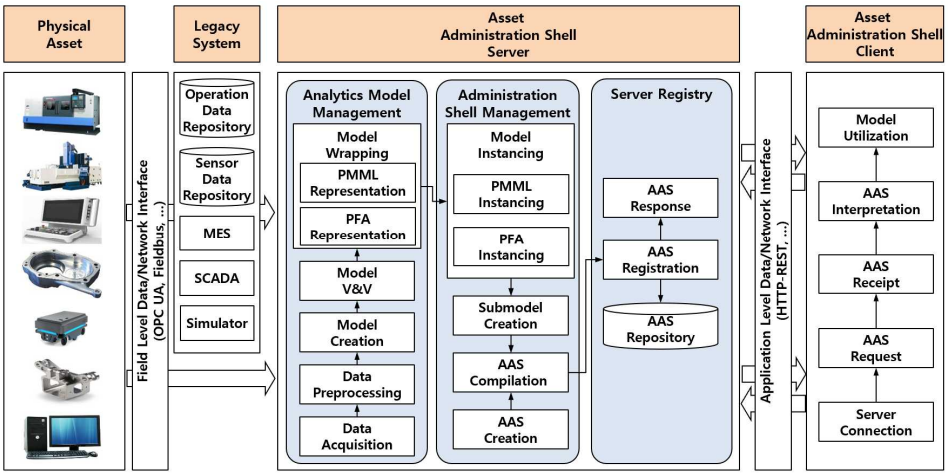


Figure 7. System Architecture

서버와 클라이언트는 Representational State Transfer(REST) API 기반 웹서비스로 연결한다. 웹서비스는 이기종 시스템 간 빠르고 유연한 상호작용을 제공하도록 서버와 클라이언트가 분리되어 있는, 즉 디커플링된(decoupled) 구조로 설계된다. 강력하게 커플링된 시스템으로는 유연성과 확장성을 보장하기 어려우므로, 자연스럽게 웹서비스 구조를 취하여 접근성을 강화하는 것이 합리적이다. 또한, 이 웹서비스 구조는 시스템 간 통신 방법 측면에서의 상호운용성을 증진시킬 수 있다. 이는 고유한 형태의 IRI · URI 기반 REST API를 이용함으로써, 클라이언트의 변경 및 교체가 이루어지더라도 동일한 REST API를 사용할 수 있다. 참고로, 자산관리쉘을 위한 소프트웨어 개발 도구(Software Development Kit : SDK)도 REST API 기반 서버-클라이언트 구조를 채택하고 있다.

4. 구 현

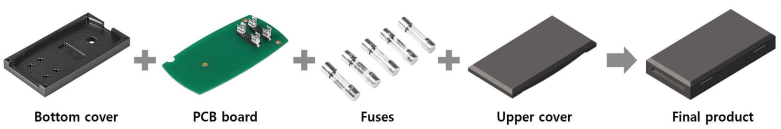
본 장에서는 제안한 방법의 실행가능성을 확인하기 위한 시제품 구현을 설명한다. 4.1절에서는 실험환경 및 구현 시나리오를, 4.2절에서는 구현 구조를, 4.3절에서는 구현 결과를 서술한다.

4.1 시나리오

시나리오의 목적은 서버 측에서 생산설비 중 하나인 열처리로(heat tunnel)를 대상으로 열처리 시간 및 온도에 대한 에너지 사용량의 PMML · PFA 예측 모델들을 생성하는 것이다. 이 모델들을 자산관리쉘과 통합하고, 클라이언트의 요청에 의해 이 자산관리쉘 인스턴스를 반환하는 것이다. 한편, 클라이언트 측에서 자산관리쉘 인스턴스를 요청하고, 서버로부터 반환된 자산관리쉘 인스턴스를 받는 것이다.

<Figure 8(a)>는 구현에 사용된 교육 · 연구용 유연 생산 시스템 장비인 Cyber-Physical Factory(이하, CP-Factory)를 나타낸다. 본 장비는 컨베이어 벨트 이송에 의하여 여러 공정 설비들을 거쳐서 간단한 시제품을 만들며(<Figure 8(b)> 참고), 공장자동화용 하드웨어와 소프트웨어로 구성되어 있다. CP-Factory에서는 PLC로부터 각 설비별 공정 시작시간과 종료시간을 포함한 운영

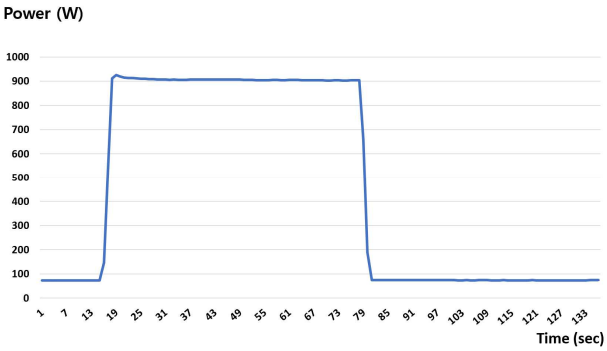
(a) Cyber-Physical Factory



(b) Product example

Figure 8. Experimental Facility

(a) Heat tunnel



(b) Power data(heating time = 55.6 sec, heating temperature = 71.2℃)

Figure 9. Target Asset and Power Data Example

데이터(operation data)를 수집하고 데이터베이스에 저장한다. 이와 동시에, 각 설비에 연결된 전력량계는 1초 단위로 전력 데이터(power data)를 측정하고, 이 전력 데이터는 데이터베이스에 저장된다. <Figure 9(a)>는 구현 대상 생산설비인 열처리로(heat tunnel)를 나타낸다. 열처리로는 열처리 시간(heating time) 및 열처리 온도(heating temperature) 설정 값만큼 제품에 열을 가하게 되는데, 열을 가하기 위한 유효전력을 사용한다(<Figure 9(b)> 참고). 이 때, x축의 시간(time)과 y축의 전력값(power)에 의해 생성되는 면적이 에너지(energy) 값이 된다.

4.2 구현 구조

<Figure 10>은 <Figure 7>의 시스템 구조에 기반한 구현 구조도를 나타낸다. CP-Factory에서 생성되는 데이터는 OPC UA를 통하여 전송되며, 운영 데이터는 마이크로소프트 Access에, 전력 데이터는 MariaDB에 저장된다. 자산관리쉘 서버와 클라이언트는 Java BaSyx SDK를 이용하여 구현하였다. Java BaSyx SDK는 자산관리쉘 개발용 Java 기반 오픈 소스 개발도구이다(BaSys 4.0, 2020). 서버와 클라이언트 통신은 웹통신 규약인 HTTP-REST를 이용하는데, 본 구현에서는 로컬호스트(localhost) 구현되었다. 데이터 애널리틱스 모델 생성에는 Weka SDK를 이용하였다. Weka SDK는 Java 기반 기계학습용 오픈 소스 개발도구이다(Waikato University, 2020). PMML 래핑을 위해서는 Java용 PMML 라이브러리인 JPMML을(Openscoring, 2019), PFA 래핑을 위해서는 Java의 범용 JSON 라이브러리를 이용하였다(Maven repository, 2020). 여기서, 서버의 기능들은 프로그래밍을 통하여 자동화하였다. 다만, 데이터 획득(data acquisition)에서는 Access와 MariaDB에서 수작업으로 추출한 CSV 파일을 사용하였다(CP-Factory의 네트워크 보안 제약에 기인).

Table 1. Factors and Levels

Factor	Unit	Level	
		-1	1
Heating Temperature	℃	34.4	55.6
Heating Time	sec	28.8	71.2

4.3 구현 결과

먼저, 중심합성계획법을 이용하여 2개의 독립변수(열처리 온도와 열처리 시간) 및 종속변수(에너지)에 대한 실험을 계획하였다. 이 경우, 13번의 실험이 실시되며, 이 때 발생되는 데이터를 학습 데이터(training data)로 사용하였다. <Table 1>은 실험 계획에서 고려된 독립변수와 수준을 정리한 것이다. 더불어, 각 독립변수의 최소값-최대값 범위 내에서 임의로 8개 쌍을 선별하여 추가 실험을 실시하였으며, 이 데이터를 테스트 데이터(test data)로 사용하였다.

CP-Factory에 연결된 제조 실행 시스템(MES4)을 이용하여 21개 제품(<Figure 8(b)> 참고)을 생산하였다. 이 때, 열처리로에서는 <Table 1>의 실험계획으로부터 도출된 열처리 온도와 열처리 시간을 각 제품별로 적용하였다. PLC에서 생성되는 운영 데이터와 전력량계에서 생성되는 전력 데이터는 OPC UA Data Hub를 통하여 각각 Access와 MariaDB에 저장된다. 구현 결과는 아래와 같다. <Figure 11>과 <Figure 12>는 구현 결과를 요약한 그림이다.

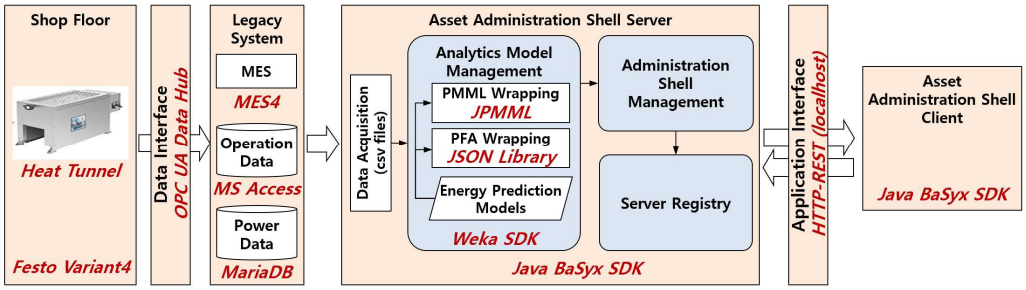


Figure 10. Implementation Architecture

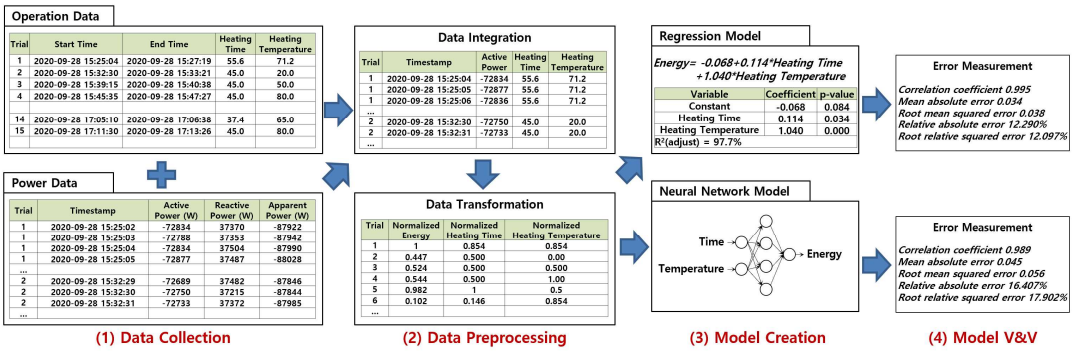


Figure 11. Implementation of Predictive Modeling

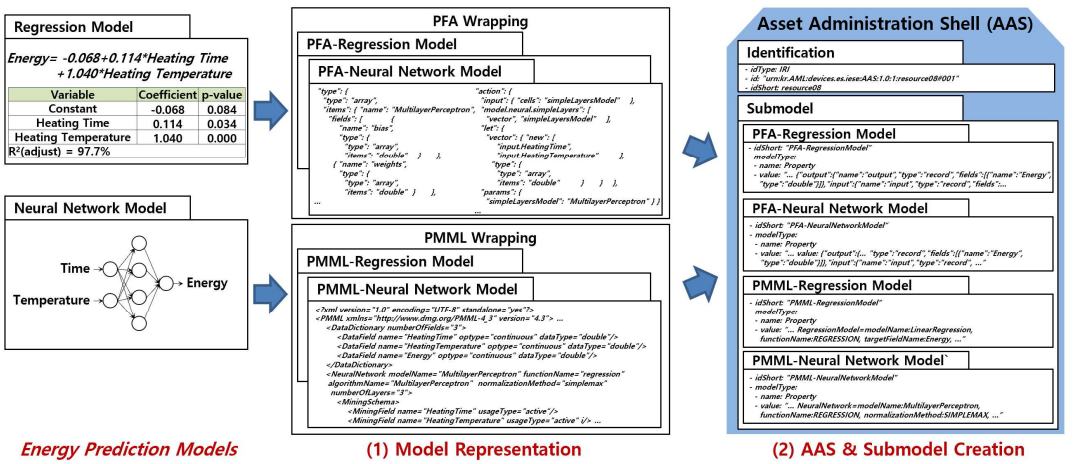


Figure 12. Implementation of Asset Administration Shell

(1) 운영 데이터와 전력 데이터 수집 : 자산관리쉘 서버에서는 운영 데이터와 전력 데이터의 CSV 파일을 읽어 들인다. Access와 MariaDB에 저장된 운영 데이터와 전력 데이터는 <Figure 11>의 (1)과 같이 구성되어 있다(운영 데이터는 주요부분만 발췌).

(2) 데이터 전처리 : <Figure 11>의 (2)와 같이, 데이터 통합 및 변환을 수행한다. 데이터 통합은 운영 데이터의 시작시간과 종료시간을 기준으로 하여, 이 범위 안에 존재하는 타임스탬프의 전력 데이터 샘플을 통합하게 된다. 그리고 데이터 변환에서는 유효전력(active power)값을 양(+)의 값으로 변환한다. 그 후, 샘플링 시간에 따른 유효전력 값으로부터 구분구적법을 이용하여 에너지 값을 계산한다. 그리고 각 변수값에 대하여 0~1 사이의 정규화를 실시한다.

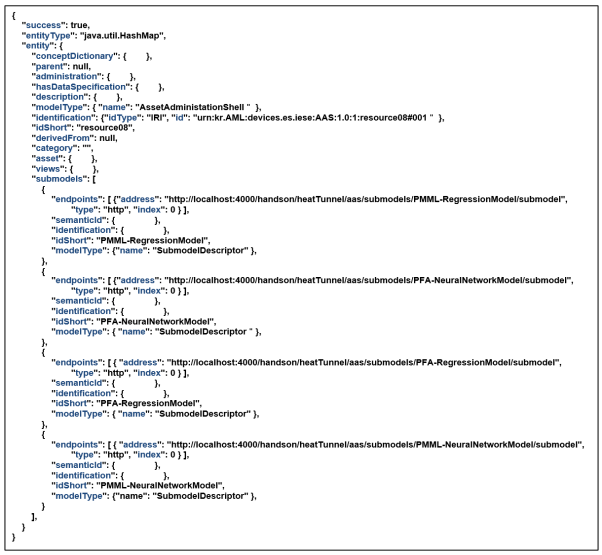
(3) 예측 모델 생성 : Weka SDK를 이용하여 예측 모델을 생성한다. 선형회귀 기법과 인공신경망 기법을 적용하여 전처리된 학습 데이터로부터 예측 모델들을 생성하는 것이다. 예측 모델은 <Figure 11>의 (3)과 같다. 인공신경망의 경우, 역전파(backpropagation) 방법을 사용하며, 은닉층은 1개로 구성되며 4개의 뉴런이 존재한다. 활성화 함수는 시그모이드, learning rate는 0.3, momentum은 0.2, epoch 횟수는 500으로 설정하였다.

(4) 모델 성능 확인: Weka SDK를 이용하여 모델의 오차를 평가한다. 테스트 데이터에 대한 예측 모델의 성능은 <Figure 11>의 (4)과 같다.

(5) PMML 및 PFA 래핑 : 선형회귀 및 인공신경망 모델에 대하여 각각 PFA와 PMML 형태로 래핑된 4개의 문서를 생성한다. <Figure 12>의 (1)은 이에 대한 일부이다. PMML의 경우는 JPMML 라이브러리를 이용하여 PMML 클래스 인스턴스 및 PMML 파일을 생성하였다. 한편, PFA는 범용 JSON 라이브러리를 이용하여 PFA 클래스 인스턴스와 PFA 파일을 생성하였다.

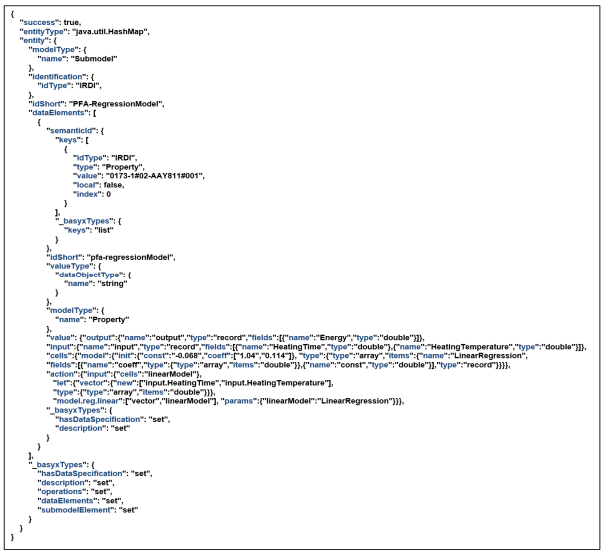
(6) 자산관리쉘 및 PMML · PFA 서브모델 생성 : 열처리로의 자산관리쉘을 생성하며, 4개의 PMML · PFA 문서를 서브모델로 생성한다. 이 문서들은 데이터 요소 내 property의 string 형태로 저장한다. <Figure 12> (2)는 PMML · PFA 모델을 탑재한 자산관리쉘을 나타낸다.

(7) 서버의 자산관리쉘 인스턴스 등록 : 서버에 자산관리쉘 인스턴스가 등록된다. <Figure 13>은 JSON으로 표현된 최상위 자산관리쉘 인스턴스 결과이다 (지면관계상 주요 컨텐츠만 발췌). 해당 자산관리쉘은 IRI 형태로 객체식별자를 가지고 있으며, 4개의 서브모델을 포함하고 있다. 서브모델별로 ID 및 URI가 할당되어 있음을 확인할 수 있다. <Figure 14>는 PFA 형태의 선형회귀 서브모델 인스턴스 결과이다 (지면관계상 다른 3개의 서브모델 인스턴스 결과는 배제).



```
{
  "success": true,
  "entityType": "java.util.HashMap",
  "entity": {
    "conceptDictionary": {      },
    "parent": null,
    "administration": {      },
    "hasDataSpecification": {      },
    "description": {      },
    "modelType": { "name": "AssetAdministrationShell " },
    "identification": {"idType": "IRI", "id": "urn:kr.AML:devices.es.iese:AAS:1.0:1:resource08#001 " },
    "idShort": "resource08",
    "derivedFrom": null,
    "category": "",
    "asset": {      },
    "views": {      },
    "submodels": [
      {
        "endpoints": [ {"address": "http://localhost:4000/handson/heatTunnel/aas/submodels/PMML-RegressionModel/submodel",
            "type": "http", "index": 0 } ],
        "semanticId": {            },
        "identification": {            },
        "idShort": "PMML-RegressionModel",
        "modelType": {"name": "SubmodelDescriptor" },
      },
      {
        "endpoints": [ {"address": "http://localhost:4000/handson/heatTunnel/aas/submodels/PFA-NeuralNetworkModel/submodel",
            "type": "http", "index": 0 } ],
        "semanticId": {            },
        "identification": {            },
        "idShort": "PFA-NeuralNetworkModel",
        "modelType": { "name": "SubmodelDescriptor " },
      },
      {
        "endpoints": [ { "address": "http://localhost:4000/handson/heatTunnel/aas/submodels/PFA-RegressionModel/submodel",
            "type": "http", "index": 0 } ],
        "semanticId": {            },
        "identification": {            },
        "idShort": "PFA-RegressionModel",
        "modelType": { "name": "SubmodelDescriptor" },
      },
      {
        "endpoints": [ { "address": "http://localhost:4000/handson/heatTunnel/aas/submodels/PMML-NeuralNetworkModel/submodel",
            "type": "http", "index": 0 } ],
        "semanticId": {            },
        "identification": {            },
        "idShort": "PMML-NeuralNetworkModel",
        "modelType": {"name": "SubmodelDescriptor" },
      }
    ],
  }
}
```


Figure 13. Implementation of Asset Administration Shell Instance



```
{
  "success": true,
  "entityType": "java.util.HashMap",
  "entity": {
    "modelType": {
      "name": "Submodel"
    },
    "identification": {
      "idType": "IRDI",
    },
    "idShort": "PFA-RegressionModel",
    "dataElements": [
      {
        "semanticId": {
          "keys": [
            {
              "idType": "IRDI",
              "type": "Property",
              "value": "0173-1#02-AAY811#001",
              "local": false,
              "index": 0
            }
          ],
          "_basyxTypes": {
            "keys": "list"
          }
        },
        "idShort": "pfa-regressionModel",
        "valueType": {
          "dataObjectType": {
            "name": "string"
          }
        },
        "modelType": {
          "name": "Property"
        },
        "value": {"output":{"name":"output","type":"record","fields":[{"name":"Energy","type":"double"}]},
        "input":{"name":"input","type":"record","fields":[{"name":"HeatingTime","type":"double"},{"name":"HeatingTemperature","type":"double"}]},
        "cells":{"model":{"init":{"const":"-0.068","coeff":["1.04","0.114"]}, "type":{"type":"array","items":{"name":"LinearRegression",
        "fields":[{"name":"coeff","type":{"type":"array","items":"double"}},{"name":"const","type":"double"}],"type":"record"}}}},
        "action":{"input":{"cells":"linearModel"},
          "let":{"vector":{"new":["input.HeatingTime","input.HeatingTemperature"],
          "type":{"type":"array","items":"double"}}},
          "model.reg.linear":["vector","linearModel"], "params":{"linearModel":"LinearRegression"}}},
        "_basyxTypes": {
          "hasDataSpecification": "set",
          "description": "set"
        }
      }
    ],
    "_basyxTypes": {
      "hasDataSpecification": "set",
      "description": "set",
      "operations": "set",
      "dataElements": "set",
      "submodelElement": "set"
    }
  }
}
```


Figure 14. Implementation of PFA-based Regression Submodel

(8) 클라이언트의 요청과 서버의 반환 : 클라이언트에서는 서버에게 자산관리쉘 인스턴스를 요청한다. 서버에서는 해당 자산관리쉘 인스턴스를 반환한다. 이러한 요청-반환을 위하여 자산관리쉘의 IRI 및 서브모델의 URI를 키 값으로 사용한다. <Figure 13>의 서브모델 URI에 접속하면 <Figure 14>와 같은 서브모델 컨텐츠를 확인할 수 있다. 서버에서 반환한 자산관리쉘 및 4개 서브모델 인스턴스와 클라이언트에서 요청한 자산관리쉘 및 4개 서브모델 인스턴스를 비교한 결과, 일치함을 확인하였다.

5. 결 론

본 논문에서는 PMML과 PFA 기반 데이터 애널리틱스 모델을 자산관리쉘과 통합하여 교환 및 공유하기 위한 방법을 개발하였다. 이를 위하여, 자산관리쉘과 데이터 애널리틱스 통합에 대한 개념 모델, 과정, 정보 구조 및 시스템 구조를 설계하였고, 시제품을 구현하였다. 본 구현에서는 서버에서의 생산설비 에너지 예측 모델 생성, 모델의 PMML 및 PFA 래핑 그리고 자산관리쉘 인스턴스 반환 과정, 클라이언트에서의 자산관리쉘 인스턴스 요청 및 수집 과정을 보여주었다.

본 논문의 의의는 이질적인 분야의 표준들을 통합함으로써 상호운용적 제조 지능화의 구현가능성을 보여준 것에 있다. 자산관리쉘은 OPC UA와 함께 RAMI4.0을 구성하는 큰 축이며, 스마트 공장의 수평적 · 수직적 통합에 필수적인 기술이다. RAMI 4.0에서의 OPC UA가 사실상 표준으로 자리매김하듯이, 자산관리쉘 또한 사실상 표준으로서의 역할이 증가할 것으로 예상된다. 이러한 상황에서, 데이터 분석 영역과 제조 영역의 표준들을 통합함으로써, 제조 지능화를 위한 데이터 애널리틱스 모델의 상호운용성을 위한 요소기술을 개발한 것이다. 다만, 본 연구는 하나의 설비를 대상으로 플랫폼 및 어플리케이션 계층을 포함하는 수직적 통합 범위의 데이터 애널리틱스 모델 상호운용성에 대한 실행가능성을 확인한 수준이다. 자산관리쉘과 PMML · PFA을 이용하여 고유 객체식별과 데이터 구조화 · 표준화를 이루었으므로, 서버-클라이언트 구조에 준하는 플랫폼 및 어플리케이션에서는 객체식별자에 근간한 정보의 접근 및 공유를 가능하게 한다. 향후, 다양한 제조사의 다양한 설비들을 대상으로 하는 수평적 통합 범위의 데이터 애널리틱스 모델 상호운용성 확보를 위해서는 후행 연구개발이 필요하다.

본 논문의 한계점으로는 자산관리쉘이라는 무궁무진한 정보 컨테이너 안에 PMML · PFA 모델만을 담아봤다는 것이며, 실험실 수준의 장비에서 하나의 설비만을 대상으로 구현이 이루어졌다. 또한, 물리적 제조 자산으로부터 수집된 데이터 자체를 자산관리쉘의 인스턴스로 통합하는 기능을 구현하지 못하였다. 그리고 클라이언트 측에서의 PMML · PFA 모델 활용 시나리오 및 구현을 제시하지 못하였다.

향후에는 다양한 설비들을 대상으로 자산관리쉘과 PMML · PFA 모델의 통합을 구현할 예정이다. 또한, 제조 자산으로부터 수집된 데이터를 자산관리쉘의 인스턴스로 내재화하는 기능을 구현할 예정이다. 데이터 자체를 자산관리쉘의 컨테이너에 담는다면, 제조 자산부터 플랫폼 및 어플리케이션 계층을 아우르는 일관적 데이터 확보와 활용이 가능하여 고수준의 상호운용성 보장이 가능하다. 이를 위하여, 자산관리쉘의 확장적 정보 모델을 포함하여 서버의 자동적 데이터 수집 · 전처리 및 자산관리쉘로의 래핑 및 등록 구현이 필요하다. 그리고 서버-클라이언트의 확장을 통하여 PMML · PFA 모델 활용 시나리오를 구현할 것이다. 구체적으로, 공장 에너지 관리 시스템(factory energy management system)에서의 에너지 저감을 위한 예측적 · 최적화 공정 계획 및 운영에 대한 시나리오다. 국소적으로는 각 설비의 자산관리쉘과 PMML · PFA 모델을 이용하여 설정된 공정파라미터에 대한 에너지를 예측함으로써 에너지 사용을 최소화하는 공정파라미터를 결정할 수 있다. 광역적으로는 제조라인의 복수 개의 설비들에 대한 자산관리쉘들을 이용하여 향후 발생할 에너지 사용량을 예측하거나, 제조라인의 에너지 저감을 위한 작업할당 및 스케줄링이 가능해 질 것이다.

참고문헌

Adolphs *et al*. (2015), Reference Architecture Model Industrie 4.0(RAMI 4.0), VDE/ZVEI Status Report.

BaSys 4.0, Java-based SDK for asset administration shell, https://wiki.eclipse.org/BaSyx (accessed on October 14 2020).

Belhadi, A., Zkik, K., Cherrafi, A., Sha'ri M. Y. and El fezazi, S. (2019), Understanding Big Data Analytics for Manufacturing Processes : Insights from Literature Review and Multiple Case Studies, *Computers & Industrial Engineering*, **137**, 106009.

Cavalieri, S. and Salafia, M. G. (2020), Asset Administration Shell for PLC representation based on IEC 61131-3, IEEE Access, **8**, 142606-142621.

Cavalieri, S., Mulé, S., and Salafia, M. G. (2019), OPC UA-based Asset Administration Shell, *45th Annual Conference of the IEEE Industrial Electronics Society*, 14-17.

Data Mining Group (2015), PFA : Portable Format for Analytics (version 0.8.1), http://dmg.org/pfa/index.html (accessed on October 8 2020).

Data Mining Group (2019), PMML 4.4-General Structure, http://dmg.org/pmml/v4-4/GeneralStructure.html (accessed on October 6 2020).

Davis, J., Edgar, T., Porter, J., Bernaden, J., and Sarli, M. (2012), Smart Manufacturing, Manufacturing Intelligence and Demand-Dynamic Performance, *Computers and Chemical Engineering*, **47**, 145-156.

Federal Ministry for Economic Affairs and Energy (2019), Details of the Asset Administration Shell : Part 1-The Exchange of Information between Partners in the Value Chain of Industrie 4.0 (Version 2.0), ZVEI Specification.

Fuchs, J., Schmidt, J., Franke, J., Rehman, K., Sauer, M., and Karnouskos, S. (2019), I4.0-Compliant Integration of Assets Utilizing the Asset Administration Shell, *24th IEEE International Conference on Emerging Technologies and Factory Automation*, 10-13.

Grangel-González, I., Halilaj, L., Coskun, G., Auer, S., Collarana, D., and Hoffmeister, M. (2016), Towards a Semantic Administrative Shell for Industry 4.0 Components, *IEEE Tenth International Conference on Semantic Computing (ICSC)*, 4-6.

Guazzelli, A., Zeller, M., Lin, W. C., and Williams, G. (2009), PMML : An open standard for sharing models, *The R Journal*, **1**(1), 60-65.

Lechevalier, D., Hudak, S., Ak, R., Lee, Y. T., and Foufou, S. (2015), A Neural Network Meta-Model and its Application for Manufacturing, *IEEE International Conference on Big Data.*

Lechevalier, D., Narayanan, A., Rachuri, S., and Foufou, S. (2018), A Methodology for the Semi-Automatic Generation of Analytical Models in Manufacturing, *Computers in Industry*, **95**, 54-67.

Lüder, A., Behnert, A. K., Rinker, F., and Biffl, S. (2020), Generating Industry 4.0 Asset Administration Shells with Data from Engineering Data Logistics, *25th IEEE International Conference on Emerging Technologies and Factory Automation*, 8-11.

Marcon, P., Diedrich, C., Zezulka, F., Schröder, T., Belyaev, A., Arm, J., Benesl, T., Bradac, Z., and Vesely, I. (2018), The Asset Administration Shell of Operator in the Platform of Industry 4.0, *18th International Conference on Mechatronics–Mechatronika*, 5-7.

Maven Repository, JSON Libraries for Java, https://mvnrepository.com/artifact/org.json/json (accessed on November 16 2020).

Ministry of Economy and Finances in France, Federal Ministry for Economic Affairs and Energy in Germany, Plattform Industrie 4.0 in Germany, Ministero Dello Sviluppo Economico in Italy, Piano Nazionale Impresa 4.0 in Italy (2018), The Structure of the Administration Shell : Trilateral Perspectives from France, Italy and Germany, International Paper.

Moghaddam, M., Cadavid, M. N., Kenley, C. R., and Deshmukh, A. V. (2018), Reference Architectures for Smart Manufacturing : A Critical Review, *Journal of Manufacturing Systems*, **49**, 215-225.

Nannapaneni, S., Narayanan, A., Ak, R., Lechevalier, D., Sexton, T., Mahadevan, S., and Lee, Y.-T. T. (2018), Predictive Model Markup Language (PMML) Representation of Bayesian Networks : An Application in Manufacturing, Smart and Sustainable Manufacturing Systems, **2**(1), 87-113.

O'Donovan, P., Bruton, K., and O'Sullivan, D. T. (2016), Case Study : The Implementation of a Data-Driven Industrial Analytics Methodology and Platform for Smart Manufacturing, *International Journal of Prognostics and Health Management*, 7026.

Openscoring, Standards-based, Open-Source Middleware for Predictive Analytics Applications, https://openscoring.io/(accessed on November 5 2019).

Park, J., Lechevalier, D., Ak, R., Ferguson, M., Law, K. H., Lee, Y.-T. T., and Rachuri, S. (2017), Gaussian Process Regression(GPR) Representation in Predictive Model Markup Language(PMML), *Smart and Sustainable Manufacturing Systems*, **1**(1), 121-141.

Park, K. T., Kang, Y. S., Im, S. J., Noh, S. D., Yang, S. G. and Kang, Y. T. (2019), Implementation of Digital Twin and Virtual Representation for Energy Efficiency Improvement of Dyeing and Finishing Industry, *Journal of the Korean Institute of Industrial Engineers*, 45(6), 491-502.

Park, K. T., Yang, J. H., and Noh, S. D. (2020), VREDI : Virtual Representation for a Digital Twin Application in a Work-Center-Level Asset Administration Shell, *Journal of Intelligent Manufacturing*, **32**, 501-544.

Pivarski, J., Bennett, C., and Grossman, R. L. (2016), Deploying Analytics with the Portable Format for Analytics(PFA), *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 579-588.

Shin, S. J. and Meilanitasari, P. (2020), An OPC UA-based Representation Method of Data Analytics Models for Interoperable Manufacturing Intelligence, *Journal of the Korean Institute of Industrial Engineers*, **46**(6), 580-592.

Tantik, E. and Anderl, R. (2017), Integrated Data Model and Structure for the Asset Administration Shell in Industrie 4.0, *Procedia CIRP*, **60**, 86-91.

Trunzer, E., Calà, A., Leitão, P., Gepp, M., Kinghorst, J., Lüder, A., Schauerte, H., Reiferscheid, M., and Vogel-Heuser, B. (2019), System Architectures for Industrie 4.0 Applications-Derivation of a generic Architecture Proposal, *Production Engineering*, **13**, 247-257.

Wagner, C., Grothoff, J., Epple, U., Drath, R., Malakuti, S., Grüner, S., Hoffmeister, M., and Zimermann, P. (2017), The Role of the Industry 4.0 Asset Administration Shell and the Digital Twin During the Life Cycle of a Plant, *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 12-15.

Waikato University, Weka libraries for Java, https://github.com/Waikato/weka-3.8 (accessed on November 2 2020).

Ye, X. and Hong, S. H. (2019), Toward Industry 4.0 Components-Insights Into and Implementation of asset Administration Shells, *IEEE Industrial Electronics Magazine*, **13**(1), 13-25.

Zhang, H., Roy, U., and Lee, Y-T. T. (2019), Enriching Analytics Models with Domain Knowledge for Smart Manufacturing Data Analysis, *International Journal of Production Research*, **58**(20), 6399-6415.

저자소개

신승준 : 고려대학교 기계공학과에서 2002년 학사, POSTECH 산업경영공학과에서 2005년 석사, 2010년 박사 학위를 취득하였다. 삼성전자, 삼성SDS, 미국 표준기술연구소(NIST) 및 부경대학교를 역임하고, 2018년부터 한양대학교 산업융합학부 부교수로 재직하고 있다. 연구분야는 사이버-물리 생산 시스템, 제조분야 빅데이터 애널리틱스, 상호운용성 및 친환경 제조이다.

이주홍 : 부경대학교 시스템경영공학부에서 2018년 학사학위를 취득하고 한양대학교 산업데이터엔지니어링학과 석사과정에 재학중이다. 연구분야는 데이터 엔지니어링, 인공지능, 스마트 팩토리이다.

박정민 : 경희대학교 산업경영공학과에서 2020년에 학사학위를 취득하고 동대학 석사과정에 재학중이다. 연구분야는 스마트 팩토리, 적층형 공정, 3D 물체인식이다.

엄주명 : 성균관대학교에서 기계공/정보통신학사를, POSTECH 산업경영공학과에서 석박통합과정을 취득하였다. 이후 스위스 로잔연방공대와 영국 캠브리지대학교에서 연구원으로 근무하였으며, 독일 SmartFactoryKL에서 선임연구원으로 근무하였다. 현재는 경희대학교 산업경영공학과에서 조교수로 재직중이다. 관심분야는 스마트팩토리, 적층형 공정, 제조인공지능, 제조데이터모델이다.