Journal of Digital Convergence
Vol. 17. No. 9, pp. 199-212, 2019

ISSN 1738-1916
https://doi.org/10.14400/JDC.2019.17.9.199

비즈니스 문서의 생산성 향상을 위한 RPA(Robotics Process Automation)적용방안에 대한 연구

현영근[1], 이주연[2*]
[1]아주대학교 산업공학과 석박사통합과정, [2]아주대학교 산업공학과 교수

A Study On The Application of RPA(Robotics Process Automation) For Productivity Of Business Documents

Young Geun Hyun[1], Joo Yeoun Lee[2*]
[1]M.D. integration process, Division of Industrial Engineering, Ajou University
[2]Professor, Division of Industrial Engineering, Ajou University

요 약 디지털화(Digitalization)가 우리의 비즈니스 환경에 다양한 변화와 혁신을 일으키고 있다. 제조업에서는 오래전부터 자동화를 위해 로봇을 활용하여 처리속도 및 품질에 혁신을 이루었다. RPA는 이러한 제조현장의 혁신을 사무공간으로 가져온 것이라고 할 수 있다. 본 연구의 목적은 사무공간에서 단순 반복적으로 이루어지는 업무에 대해 생산성을 향상시키는 것을 그 목적으로 한다. 이러한 생산성 향상과 관련하여, 비즈니스 자동화(Business Automation)에 대한 개념을 살펴본 후, 비즈니스 문서 작업과 관련하여 자동화의 가능성을 확인하기 위해 5가지 업무영역을 대상으로 애자일 방법론을 활용하여 시뮬레이션을 수행하였다. 결론적으로, 품질점검 관련 97.3%, 편집 디자인 관련 31.7%의 생산성 향상이 가능함을 확인하였으며, 실제 업무에 적용하기 위한 방향성에 대해서도 살펴보았다. 향후 연구에서는 이러한 결과를 바탕으로 IPA(Intelligent Process Automation)의 적용방안에 대해 진행하고자 한다.

주제어 : 비즈니스 자동화, RPA, 문서 자동화, 디지털 노동, 인공지능

Abstract Digitalization is creating a variety of changes and innovations in our business environment. In manufacturing, robots have long been used for automation to innovate processing speed and quality. The RPA brings these innovations in manufacturing sites to the office space. The purpose of this study is to improve productivity for simple, repetitive tasks in these office space. For identify the potential of automation related to productivity improvement, I looked at the concept of business automation, and then simulated the five areas of business documentation works with agile methodology. In conclusion, I confirmed that productivity improvement of 97.3% in quality inspection and 31.7% in editorial design is possible, and examined the direction to apply to actual work. Based on these results, future study will explore the application of Intelligent Process Automation (IPA).

Key Words : Business Automation, RPA, Document Automation, Digital workforce, Artificial Intelligence

*This work was supported by Ministry of Trade, Industry and Energy(MOTIE) and Korea Institute for Advancement of Technology (KIAT) of the Republic of Koreaunder Grant (N0001083).
*Corresponding Author : Joo-Yeoun Lee(jooyeoun325@ajou.ac.kr)
Received June 17, 2019 Revised July 20, 2019
Accepted September 20, 2019 Published September 28, 2019

1. 서론

RPA(Robotic Process Automation)는 사람이 수행하는 업무 중 단순 반복적이며, 패턴화된 작업을 컴퓨터가 자동적으로 처리할 수 있도록 전환하는 것을 말한다. 여기서 로보틱(Robotic)이란 물리적 로봇이 아니라, 사람이 하는 인지적인 일을 대신한다는 뜻에서 '컴퓨터 프로세스'를 말한다. RPA 목적은 사람이 하던 低부가가치 업무를 자동화함으로써, 高부가가치 및 창의적 업무에 인력을 집중할 수 있도록 해주는 것이다[1,2].

이러한 RPA의 장점으로 많은 기업들이 생산성 향상을 위해 Value Chain 전반에 걸쳐 RPA를 적용하고 있다. 이것은 생산, 영업, 구매, 재무 및 인사 등 모든 영역의 다양한 업무를 RPA로 대체할 수 있다는 것을 의미한다[3]. 그중 가장 활발하게 RPA를 도입하는 분야는 금융권으로, 챗봇(Chat-Bot)을 활용하여 단순 반복적 업무의 자동화뿐만 아니라 고객응대 그리고 개인 맞춤형 로보어드바이저(RoboAdvisor)를 활용한 고객서비스 분야에도 활용하고 있다[4,5]. 또한 RPA의 접목이 어려울 것으로 생각되던 의료계 및 법률계도 RPA를 적극적으로 도입하여 소기의 성과를 내고 있다[6].

이렇듯 다양한 분야에서 생산성을 향상시키기 위해 RPA를 도입하고 있으나, 화이트컬러의 핵심 업무인 비즈니스 문서작업과 관련한 생산성 향상에는 상대적으로 소홀하고 있다. 그 이유는 비즈니스 문서작업은 인간의 인지적 판단이 필요한 업무라 로봇적용이 불가능하다고 생각하기 때문이라 예상된다.

비즈니스 문서의 목적은 시간적 그리고 공간적으로 떨어져 있는 구성원 및 관련자 간 정보공유라고 할 수 있다. 특정 비즈니스 문서는 창의적 아이디어와 인간의 인지적 판단을 기반으로 작성되는 문서가 있는가 하면, 또 다른 문서는 참조하는 문서들을 일부 재활용하여 새로운 문서를 만드는 경우도 있다. 이렇게 다른 문서를 참조하여 작성하는 경우, 컨텐츠를 새로운 문서의 작성목적에 맞게 현행화(예: 목자구조, 용어/단어 수정 등)하는 단순 반복적인 작업이 발생하게 된다. 이때 휴먼 에러(Human Error)가 빈번히 발생하여, 결국 문서의 품질을 떨어뜨리는 역할을 하게 된다. 만약, 이러한 패턴을 보이는 문서작업에 RPA를 도입한다면, 업무생산성 및 품질 향상뿐만 아니라, 사회적으로도 큰 파급효과가 있을 것으로 예상된다.

본 논문에서는 비즈니스 자동화(Business Automation)에 대한 개념을 설명하고, RPA 부상 배경 및 관련 솔루션, 그리고, 디지털 노동(Digital Labor)이 인간노동(Human Labor)을 대체하는 다양한 사례를 분석하였다. 이를 바탕으로 비즈니스 문서에서 Robot을 적용할 수 있는 문서유형을 정의하고, 이에 따른 자동화 적용가능 범위 그리고 구체적 In-House 개발방안을 살펴보고자 한다. 마지막으로 각 적용대상 별 시뮬레이션을 통해 생산성이 얼마나 향상될 수 있는지 연구하고자 한다.

2. 비즈니스 자동화 개념

글로벌 주요 기업은 지속적인 비용절감을 위해 1990년대 ERP에 이어 2000년 이후 Offshore BPO에 주력하였으나, 최근 점차 SW-Robot을 이용하기 시작했다. 이렇게 ERP에서 BPO로, 그리고 RPA로 변화를 꾀하는 가장 큰 이유는, 여전히 많은 인력이 불필요한 단순 업무에 너무나 많은 시간을 소모함으로써 업무 효율성 개선에 있어서 한계에 직면했기 때문이다. (Biz Operation에서 단순 업무 비중이 70%에 달하며, 45%는 자동화를 통해 U$2조의 비용절감이 가능할 것으로 예상, PwC, 2016) 결국, 이러한 과정은 기존 ERP 중심의 '프로세스 혁신'을 넘어, '프로세스 자동화'로 개념이 발전됨을 의미한다[6,7].

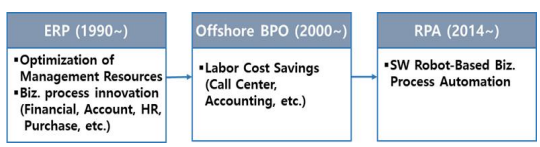

Fig. 1. Automation trend of global company

2.1 RPA의 부상

빅데이터, 인공지능(AI) 등 4차 산업혁명의 다양한 기술들이 기업경영 전반에 활용되는 과정에서 인간의 노동을 디지털 노동(Digital Labor)이 대체하고 있다. 이것은 자연언어로 소통하는 새로운 노동 형태인 디지털 노동(Digital Labor)이 부상한다는 것을 의미할 수 있다[1].

독일 취리히보험(Zurish Insurance)은 RPA를 도입해 생산성을 크게 향상시킨 대표적 사례이다. 2014년만 해도 이 회사 직원들은 보험금을 지급하기 위해 고객들이 제출한 병원진단서, 교통사고 내역서 등 수십 가지의 서류를 직원들이 직접 입력해야 했으며, 이 업무를 담당하는 81명의 직원 중 37명이 단순 반복적인 업무에 매달려야 했다. 취리히 보험은 이러한 문제를 해결하기 위해 RPA를 도입하였으며, 보험 계약확인, 보상금 지급 등 51

개 프로세스, 총 27명분의 업무를 로봇이 대신 처리하게 하였고, 이 인력들은 고객 만족도가 높아지는 서비스 품질업무에 투입하여 부가가치를 높이는 역할을 하였다.[3]

이러한 RPA를 활용한 비즈니스 자동화가 초기에 안착할 수 있었던 것에는 RPA 도구가 큰 역할을 하였다. 미국 시장조사 기관인 Forrester^{TM}는 전 세계에 RPA 솔루션을 납품하고 있는 12개 대표 벤더의 기능을 6가지 기준으로 분석한 결과, UiPath, Blue Prism, Automation Anywhere의 3개사 제품이 시장에서 Leading 솔루션으로 확인되었다. 하지만, 이 6가지 기준에 해당하는 숫자가 높다고 해서 모든 비즈니스에서 반드시 좋은 솔루션을 의미하는 것은 아니며, 적용하고자 하는 조직의 특성과 전략을 고려하고 벤더의 특징을 비교하여 상세한 제품평가를 한 후 요구에 맞게 가중치를 조정하여 선택하는 것이 필요하다[8,9].

	Forrester's weighting	Automation Anywhere	Blue Prism	Contextor	EdgeVerve Systems	Kofax	Kryon Systems	NICE	Pegasystems	Redwood Software	Softomotive	UiPath	WorkFusion
Current Offering	50%	3.69	3.39	2.54	3.03	2.33	3.02	3.51	2.99	2.58	2.14	3.53	3.09
Bot development and core functions	20%	3.70	2.50	3.10	2.85	1.65	3.45	3.60	3.25	2.35	2.85	3.25	2.30
Control room, system management, reporting, and resilience	10%	2.80	3.80	2.25	3.25	2.75	3.25	3.25	2.45	3.50	2.85	3.80	3.45
RPA analytics	10%	3.66	2.00	1.66	3.34	2.00	2.33	2.33	3.00	1.34	2.00	3.66	3.00
Architecture	10%	4.33	3.66	3.00	3.67	2.67	3.66	4.33	3.34	2.67	1.99	3.99	2.99
Breadth of use case	10%	4.10	3.40	3.00	2.35	1.90	2.05	2.00	2.65	1.60	2.25	2.75	2.15
Deployment, governance, and security	40%	3.66	4.00	2.33	3.00	2.67	2.99	3.99	2.98	3.00	1.66	3.66	3.68
Strategy	50%	4.25	4.25	2.25	2.50	2.75	2.75	2.50	3.25	2.75	2.25	4.00	3.25
Vision, execution, and strategy	100%	4.25	4.25	2.25	2.50	2.75	2.75	2.50	3.25	2.75	2.25	4.00	3.25

Fig. 2. Six Comparison Criteria of RPA Solutions Functions

2.2 디지털 노동(Digital Labor)의 스펙트럼

많은 기업들이 생산성 향상을 위해 Value Chain 전반에 걸쳐 RPA를 활용하여 자동화를 추진하고 있다. 이것은 생산, 영업 및 구매, 재무, 인사, IT 등 모든 영역의 다양한 업무를 RPA로 대체할 수 있다는 것을 의미한다. 예를 들어 데이터 입력, e메일 수신/발신, 리포트 작성 등의 업무는 물론 전산 시스템과 연동된 매출 보고서 작성, 시장 동향 수집 등의 업무도 수행할 수 있다[3].

현재, RPA가 가장 널리 쓰이고 있는 분야는 금융권이다. 은행의 비대면 고객대응 분야 및 보험사의 고객/계약관리 등에 적용되면서 금융회사는 20~30% 비용절감 효과를 누리고 있다. 또한 단순 반복되는 문의에 답해주는 '챗봇' 및 개인 맞춤형 로보어드바이저(RoboAdvisor) 등의 고객서비스 분야(예: 퇴직연금 설계, 투자 포트폴리오, 펀드 등) 뿐만 아니라 자금세탁 모니터링이나 사이버보안 등 사실상 전 분야에 걸쳐 활용하고 있다. 특히 로보어드바이저는 기업 내부 데이터를 바탕으로 외부 유통 정보의 자연어처리(NLP, Natural Language Processing)와 사회인식 알고리즘을 적용하여, 투자의사 결정 및 자산운영 분야에 활용하고 있다[4,5]

또한 인공지능(AI, Artificial Intelligence) 기반의 소프트웨어 로봇과 연계하여 전문가 업무영역의 생산성향상 및 서비스 품질제고를 위한 혁신을 도모하고 있다. 예를 들어, 美 앤더슨 암센터는 암진단에 SW-Robot을 활용하여 일반 의사의 암진단 오진율 20%를 대폭 낮추었으며(대장암 98%, 방광암 91%, 췌장암 94%로 진단 정확도 제고), 美 로펌 Baker & Hostetler는 Ross라는 SW-Robot을 활용하여 변호사 업무 중 가장 많은 시간이 들어가는 판례 분석(전체 업무 30% 수준)을 대체하였다. 국내 법률회사인 헬프미 또한 기존 30만~40만원의 비용이 들던 고객 채무소송 소장 작성을 자동화하여 90% 절감된 3.9만원에 제공하고 있다[6].

이렇듯 분야와 업무형태를 구분하지 않고, 패턴화되고 규칙적인 업무에 대해 Robot을 활용하여 업무생산성을 지속적으로 향상하고 있으며, 이것은 사람의 행동을 최소화하는 RPA(단순반복 업무의 자동화)에서 사람의 판단을 최소화하는 IPA(非 패턴화된 업무로의 확대)로 발전하고 있다[6].

Table 1. Comparison of RPA(Robotic Process Automation) and IPA(Intelligent Process Automation)

Category	RPA	IPA
Concept	Mimic Human Action	Mimic Human Judgement
Application area	Routine, Repetitive & Rules-Based	Non-Routine Pattern Recognition etc.
Major Role	Follow Instruction	Come to Conclusion
Market Status	Mature	Emerging
Cost of construction (period)	Relatively Low(Weeks)	Relatively High(Months)

3. 비즈니스 문서의 자동화 적용방안

앞에서 살펴본바와 같이, 금융권을 중심으로 한 단순 반복적 업무처리의 자동화뿐만 아니라 RAP의 접목이 어

려운 분야라 생각되었던, 의료계 그리고 법률계도 RPA를 적극적으로 도입하여 업무생산성을 향상시키는 노력을 하고 있다. 하지만, 사무환경에서 지속적으로 발생하는 비즈니스 문서와 관련해서는 자동화 적용이 미진한 것이 현실이며, 이를 자동화하였을 때 사회적으로 효과가 매우 클 것으로 예상된다.

3.1 RPA 관련 선행연구 분석

4차 산업과 관련된 연구는, 각 핵심 요소기술(e.g. AI, loT, Robot, Blockchain 등)에 대한 성숙도 및 트렌드 등에 집중하여 기술하는 경향이 있다[10]. 이것이 의미하는 것은 실제 비즈니스 환경에 어떻게 적용할 것인지, 그리고 적용하였을 때 어떻게 비즈니스 환경을 개선할 수 있을지에 대한 연구가 다소 부족하다는 것을 의미할 수 있다. 특히, 본 연구 주제인 RPA 및 비즈니스 자동화(Business Automation)와 관련된 연구는 그 사례를 찾아보기 어려우며, 그나마 이미 적용된 사례에 대한 설명만 제시할 뿐 새로운 방향성을 제시하지는 못하고 있다[11-13].

3.2 비즈니스 문서의 유형분석

비즈니스 문서는 시간적 그리고 물리적으로 떨어져 있는 구성원 및 관련자간 정보공유를 목적으로 작성된다. 이러한 비즈니스 문서는 일반적으로 그 유형에 따라 일정한 형식 그리고 컨텐츠 흐름상 유사한 패턴을 보이며, 크게 세 종류, 즉, 보고서, 기획서 및 제안서로 구분된다.

■ **보고서 (Report)**

기업내 특정 업무에 대한 진행상황 또는 분석대상 과제의 결과를 직책자에게 보고 시 작성되는 문서로, 개인의 의견보다는 Fact를 중심으로 현재의 상황에 대한 내용으로 작성된다.

■ **기획서 (Planning Document)**

특정 업무를 추진하기 전, 향후 계획을 수립하는 경우 작성하는 문서로, Fact 기반 예측을 중심으로 작성한다. 마케팅 계획 및 추진계획서 등이 이에 해당된다.

■ **제안서 (Proposal Document)**

고객사의 제안요청서(RFP, Request For Proposal)에 대해 어떻게 문제를 해결할 수 있는지 구체적 방안을 제시할 때 작성되는 문서로, 고객 Sales 활동의 문서를 의미한다.

비즈니스 문서의 공통점은 인간의 창의적 아이디어와 인지적 이해 그리고 상황에 대한 판단을 기반으로 작성된다는 것이다. 하지만, 작성방식 및 패턴을 살펴보면 Table 2.와 같이 그 차이는 명확하다.

Table 2. Comparison of business documents

Category	Report	Planning Document	Proposal Document
Number of pages	1 ~ 10 Pages	1 ~ 20 Pages	100~3,000 Pages
Number of Workers	1~ 3 People	1 ~ 3 People	3 ~ 50 People
Contents	Status Information	Future Plans	Suggesting Plans
Method of Writing	Create new Document		Write new document with reference
Method of Quality Check	Logical Validity		Document Standard Compliance

※ 일반적인 특징이며 상황/목적에 따라 상이할 수 있음

보고서와 기획서는 페이지 수, 인력 수, 컨텐츠, 작성방식 그리고 품질점검 방식에서 비슷한 특징을 보이나, 제안서는 문서작성 패턴 및 방식에 있어서 큰 차이를 보이고 있다. 무엇보다 가장 큰 차이점은 작성방식에 있으며, 보고서와 기획서는 현황분석을 통해 획득한 Fact를 기준으로 새로운 문서를 작성하는 반면에, 제안서의 경우 기존 참조문서를 활용하여 작성한다는 것이다. 따라서 참조하는 기존 문서에서 수정해야 할 단어/용어 뿐만 아니라 작성표준에 벗어난 사항들이 많이 존재한다. 예를 들어, 폰트 유형, 표 양식, 도식 스타일, 전반적인 색감 및 목차점검 등이 이에 해당된다. 또한 상대적으로 많은 사람들이 동시에 작업함에 따라 각자 상이한 스타일로 작성하여 문서의 일관성 확보를 위해 수정할 부분도 매우 많이 존재한다. 따라서 이러한 비표준과 오류를 수정하기 위한 품질점검 단계가 별로도 존재하며, 특히 디자인적으로 일관성을 확보를 위해 편집 디자이너가 작업하는 단계 또한 별도로 존재한다. 문제는 이러한 '품질점검 단계' 및 '편집 디자인' 작업이 문서의 내용적인 측면에서의 품질향상보다는, 대부분 단순 반복적 작업을 중심으로 한 패턴적용 측면의 품질향상이라는 것이다.

결론적으로, 비즈니스 문서 중 제안서에서 단순 반복적인 작업 패턴을 보이는 업무가 가장 많이 존재하여 낮은 업무생산성을 보이고 있으며, 또한 잦은 휴먼에러(Human Error)로 인해 문서품질을 떨어뜨리는 형상이 발생함에 따라, Robot 기반의 RPA를 적용했을 때 가장 효과가 클 것으로 예상된다.

3.3 비즈니스 문서의 자동화 영역 선정

앞에서 언급하였듯이, 인간의 인지적 판단보다는 일정

한 패턴에 의한 단순 반복적 작업이 자동화 대상이 될 수 있으며, 제안서 작업 측면에서는 이에 해당하는 부분이 Table 3.과 같이 5가지 영역이 가능할 것으로 예상되다. 이 5가지 영역에는 크게 두 가지 특징이 존재하며, 첫 번째 특징은, 제안서를 작성하는 영역보다는 제안서를 작성한 이후의 품질점검 활동에 자동화 기능이 가능하다는 것이며, 그 원인은, '품질활동'은 오류를 찾아내고 수정하는 활동에는 사람의 생각과 인지적 판단보다는 정해진 패턴에 의한 반복적 작업이기 때문이다. 두 번째 특징은 '①오탈자 검증 자동화' 및 '②목차검증 자동화' 영역은 제안서를 작성하는 구성원에게, 그리고 '③Text 표준적용 자동화', '④Table 표준적용 자동화', '⑤선/화살표 표준적용 자동화' 및 '⑥작업 전후비교 자동화'는 편집 디자이너에게 유용한 자동화 영역이라는 것이다.

Table 3. Automation Area and Plan

Automation Area	Automation Plan
①Typographical Error Verification	▪Automatic extraction of typographical Error and non-registration words by MSDN Standard Dictionary ▪Bulk editing of the same typographical and non-registered words in the document
②Table of Contents Verification	▪Automatic check of the table of contents described on each page -Automatic extraction of the table of contents - Automatic checking of the numbering sequence and whether the table of contents name is duplicated
③Text Standard Application	▪Applying specified standards -Automatically check whether standard is applied (eg font, color, size etc) -Automate modification according to standard
④Table Standard Application	▪Applying specified standards -Automatically modify table lines (thickness, solid / dotted line, color) -Automate the editing of text and background color in each table cell
⑤Line/Arrow standard application	▪Applying specified standards -Automatic check whether the line standard is applied -Automate editing of line (thickness, line, color, head)

4. 자동화 개발방안

4.1 자동화 Robot 구현방안

대표적인 문서작성 도구인 MS Office(PowerPoint)는 .NET Framework 기반에서 C# Language로 개발되었으며, 이를 컨트롤하기 위해 자동화 Robot은 C# Language로 개발하였다.

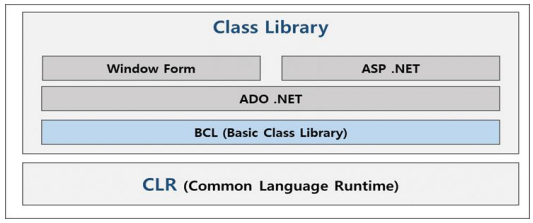

Fig. 3. Component of .NET Framework

다른 프로그램 Language와 다르게 C#에서는 CLR(Common Language Runtime) 모듈을 제공하고 있으며, 이것은 Visual Studio에서 작성된 Source Code의 exe파일을 운영체제와 무관하게 동작할 수 있도록 재컴파일하는 역할을 한다. 이것이 가능한 이유는 .NET 프로그램 상 exe파일이 완전한 기계어가 아닌 IL(Intermediate Language, 중간언어)이기 때문이며, 결국 운영체계에서 MS Office과 같은 응용 프로그램을 컨트롤할 수 있도록 해주는 것이다.

본 논문의 목적인 제안문서의 품질점검 업무생산성 향상을 위한 Robot 개발을 위해서 BCL (Basic Class Library) 및 PowerPoint API를 사용하였으며, 실행결과를 직접 확인하기 위해 콘솔보다는 Window Form을 활용하여 검증하였다. 다만, Web Class Library 인 ASP.NET와 데이터베이스 Class Library인 ADO.NET은 그 목적에 부합되지 않아 사용하지 않았다. 또한, PowerPoint를 컨트롤할 수 있는 API는 Microsoft의 MSDN (MicroSoft Developer Network)에서 제공하는 예제 소스 및 활용방법을 참조하였다.

4.2 개발방법론 적용방안

문서자동화 Robot은 요건에 대한 정확한 기준과 사례가 없으므로 기존 Waterfall 방식의 절차적인 개발방법론보다는, 각 단계를 병렬적으로 진행하여 보다 민첩하게 개발할 수 있는 애자일(Agile) 방법론을 기준으로 개발하였다.

애자일 방법론은 진행되는 프로젝트의 특성과 구성원의 참여도 등 실제 프로젝트 여건에 따라 선택할 수 있으며, XP (eXtreme Programming), Scrum, FDD (Feature Driven Development), DSDM (Dynamic System Development), Lean Software Development, Crystal Clear, ASD(Adaptive Software Development)등이 그 예이다[7,14]. 이 중 가장 많이 사용되는 방법론인 Scrum을 적용하였으며, 이 방법론은 Product Backlog를 기준으로 Sprint는 하루 8시간씩 근무하여 30일의 반복적인 개발주기를 갖으며, Sprint Planning Meeting에서는 요구사항에 필요한 각각의 작업

목록을 도출(Sprint Backlog)하고 매일 진행되는 진척 사항 및 이슈를 공유하는 회의(Daily Scrum Meeting)를 진행하는 것이 일반적이다[14].

문서자동화 Robot을 개발하는데 있어서, Product Backlog는 Table 3.를 기준으로 하였으며, 5개 자동화영역 각각 하나의 Sprint로 설정하여 Daily Scrum Meeting을 통해 업무생산성 향상을 위한 개발방안을 협의하였다. 다만, 각 자동화영역을 개발하는데 어느 정도 시간이 소요될지 예측이 불가능하여 Sprint의 기간을 설정하진 않았다.

4.3 기능별 자동화 Robot 개발방안

① 오탈자 검증 자동화

제안서에 기술된 단어를 MS-Office Dictionary에 등록된 표준어와 비교하여 오탈자를 일괄적으로 검사하는 것으로 MS-Word Object를 사용하여 개발하였다.

```
    private List<string[]> lstErrorSpellCheckList_ = null;

    // Object declaration of Microsoft Word
    Word.Document objWordDoc = null;
    // Collection object declaration
    Word.ProofreadingErrors collSpellingError = null;

    // first string array about spell error
    string[] aryErrorData = null;
    // List object of the spell error
    List<string[]> lstTemp = new List<string[]>();
    // Temporary variable of the grammatical error
    int iErrorCnt = 0;

    // Repetition by number of slide
    foreach(PowerPoint.Slide slide in
this.PPTPresentaion_.Slides)
    {
      iSlidePageNo++; // the counter of slide page
      // The number of shapes of selected slide
      shapesCnt =
PPTPresentaion_.Slides[iSlidePageNo].Shapes.Count;
      for (int s = 1; s <= shapesCnt; s++) //The number of
shape
      {
        // New document creation in MS Word collection
        objWordDoc = WordApp_.Documents.Add(ref
objTemplate,
          ref objNewTemplate, ref objDocumentType,
          ref objVisible);
        // Insert text in Range object collection

objWordDoc.Words.First.InsertBefore(sbTemp.ToString());
        // Setting ProofreadingErrors collection
        collSpellingError = objWordDoc.SpellingErrors;
        iErrorCnt = collSpellingError.Count;

       // Add the words identified as spelling error to list object
        for (int i = 1; i <= iErrorCnt; i++)
        {
          aryErrorData = new string[2];
          aryErrorData[0] = iSlidePageNo.ToString(); // slide
page
          aryErrorData[1] = collSpellingError[i].Text.ToString();
         // Discovered Spell Error Words

          lstErrorSpellCheckList_ = lstTemp.Add(aryErrorData);
        }
      }
    }
```

Fig. 4. Source Code(Searching Part) of Typographical Error Verification Automation

위와 같이 확인된 오탈자는 단순 실수에 의해 발생하는 경우도 있으나, 여러 구성원이 동시에 작업을 진행함으로써 발생되는 오류도 상당히 많이 존재한다. 대표적으로 이음동의어가 이에 해당하며, 예를 들어 고객사가 '행정안전부'라고 할 때, '행안부', '행정부', '행정 안전부', 'MOIS' 또는 '고객사' 등으로 각각 다르게 기술하는 경우를 의미하며, 이러한 유사어들을 전체 문서(여러 파일)에 걸쳐 일괄적으로 수정하여 단순업무를 최소화할 수 있도록 개발하였다.

```
    // Temporary variable for string processing
    StringBuilder sb = new StringBuilder();
    // The variable declaration of string
    StringBuilder sbChange = new StringBuilder();
    // The string variable declaration of a similar word
    string[] arySearch = null;

    sbChange.Remove(0, sbChange.Length);
    // Setting up a word to replace similar words with one word
    sbChange.AppendFormat(@"{0}", tbChangeWord.Text);

    // Repeat by Slide Count
    foreach (PowerPoint.Slide slide in
        this.PPTPresentaion_.Slides)
    {
      // Repeat by Shape Count
      foreach (PowerPoint.Shape shape in slide.Shapes)
      {
        if ((MsoShapeType.msoPlaceholder == shape.Type) ||
          ((MsoShapeType.msoPlaceholder != shape.Type) &&
          (MsoTriState.msoTrue == shape.HasTextFrame)))
        {
          arySearch = tbSearchWord.Text.Split(',');
          // String array creation by a similar string
          iAryCnt = arySearch.Length;

          for (int s = 0; s < iAryCnt; s++)
          {
            if (0 >= shape.TextFrame.
                    TextRange.TrimText().Length)
            {
              continue;
            }
            // Replace as a word to replace similar words
            PPTTextRange_ = shape.TextFrame.TextRange;

            PPTTextRange_.Replace(arySearch[s].Trim(),
              sbChange.ToString(), 0, MsoTriState.msoFalse,
              MsoTriState.msoFalse);
```

```
            }
            // Clean-up code
            arySearch = null;
        }
    }
}
```

Fig. 5. Source code(Change of similar words) of Typographical Error Verification Automation

② 목차 검증 자동화

제안서의 목차는 항상 문서 상단(Header 영역)에 위치하며 정해진 패턴(숫자+문자, 예: '1.2.2 구성방안')으로 작성된다는 특징이 있다. 따라서 특정위치에 있는 Input Box를 확인한 후, 텍스트를 읽어 각 페이지의 "숫자"가 순차적으로 나열되는지, 그리고 연결된 페이지 상에서 "문자"가 동일(중복) 한지를 점검하는 방식으로 목차검증 로직을 수립하였다.

목차검증을 위해 먼저 고려해야 할 것은, 목차의 패턴을 정의해야 하며, 본 시뮬레이션에서는 가장 기본적인 패턴인 『1 > 1.1 > 1.1.1 > 가 > 1) > 가)』를 기준으로 수행하였다.

```
private string[] GetPattern(string strTextData)
{//String-array declaration that returns analyzed pattern information
    string[] aryPatternRetVal = null;
     //String-array declaration for storing pattern expressions
    string[] aryPatternExpressions = new string[] {
        @"^[0-9]{1,}\.[0-9]{1,}\.[0-9]{1,}\.*", // 1.1.1
        @"^[0-9]{1,}\.[0-9]{1,}\.*", // 1.1
        @"^[0-9]{1,}\)\.*",  // 1) <== Parentheses ) pattern
        @"^[0-9]{1,}\.\ *",          // 1
        @"^[가-하]{1}\)\.*\ *",      // 가) pattern
    };
     // String-arrary declaration for storing pattern types
    string[] aryPatternType = new string[] {
        @"N.N.N",    // 1.1.1
        @"N.N", // 1.1
        @"N)",  // 1) <== Parentheses ) Pattern
        @"N",    // 1
        @"K1)", // 가) pattern
    };
     //  String-array declaration for "numbering" type
    string[] aryChars = new string[] {
        @"가", @"나", @"다", @"라", @"마", @"바", @"사",
        @"아", @"자", @"차", @"카", @"타", @"파", @"하",
    }
}
```

Fig. 6. Source Code(Pattern Declaration Part) of Table of Contents Verification Automation

두 번째로 고려해야 할 사항은, 파일들의 각 페이지에 기술된 목차를 읽어오는 것이다. 시뮬레이션 개발 시, 목차는 각 페이지 상의 특정한 위치에, 그리고 정확한 패턴에 의해 작성된다는 사실에 기반하였으며, 목차를 읽어올 때 "목차수준(Depth)+텍스트"의 형식으로 작성되었는지 확인하며 배열에 저장하였다.

```
private void ReadTextSlide(ref int iSlideDirection,
    ref int iShapeTotalCnt, ref SortedDictionary<float, string>
    sdOverviewList)
{
    // Temporary object for archiving shape objects
    PowerPoint.Shape pptTempShape = null;
    StringBuilder sbText = new StringBuilder();
    StringBuilder sbSaveData = new StringBuilder();
    // StringBuilder object use for speed improvement
    string strContentsLevel = string.Empty;
    // Table of content level
    float fPosition = 0.0f;//Setting the current table of contents
    float fKeyPostion = 0.0f;//Setting the new table of contents

    // The case of newly read table of contents is below
    for (int iShapesCnt = 1; iShapesCnt <= iShapeTotalCnt;
iShapesCnt++)
    {
      pptTempShape = PPTShapesObject_[iShapesCnt];
      // Create object instance to control shape in Slide
      fKeyPostion = 0.0001f;  // The level in table of content

      if ((MsoShapeType.msoPlaceholder ==
          pptTempShape.Type) ||
          ((MsoShapeType.msoPlaceholder !=
          pptTempShape.Type) && (MsoTriState.msoTrue ==
          pptTempShape.HasTextFrame)))
       {
        // Read title area in document (each page)
            if (((int)MsoOrientation.msoOrientationVertical ==
            iSlideDirection)   &&   (pptTempShape.Top   <
        iHorizontal_Top_01_) ||
            (((int)MsoOrientation.msoOrientationHorizontal ==
            iSlideDirection) && ((pptTempShape.Left <
            iVertical_Left_01_) && (pptTempShape.Top <
            iVertical_Top_03_)) || (pptTempShape.Top <
            iVertical_Top_02_)))
          {
            sbText.Remove(0, sbText.Length);
            sbText.AppendFormat(@"{0}",
            pptTempShape.TextFrame.TextRange.Text.Trim());

            // Verify whether text structure is table of contents
pattern
            if ((2 < sbText.Length) && (true ==
               IsItTableContentsFormat(sbText.ToString())))
            {
               strContentsLevel =
               GetLevel(sbText.ToString().Split(' ')[0]);
               // Setting the table of contents level
               fPosition=float.Parse(strContentsLevel.
               ToString().Split('^')[0]);
               // Setting to a numeric value for table of
contents level

               // Extracting strings from the table of contents
               if (MsoShapeType.msoPlaceholder ==
                    pptTempShape.Type)
               {
               if (2 < sbText.Length)
               {
                  sbSaveData.Remove(0, sbSaveData.Length);
                  sbSaveData.AppendFormat(@"{0}^{1}",
                  sbText.ToString(), strContentsLevel);

                  SaveDuplicateKey(fPosition, fKeyPostion,
                  sbSaveData.ToString(), sdOverviewList);
               }
            }
          }
       }
    }
}
```

Fig. 7. Source Code(Extraction & Pattern Analysis Part) of Table of Contents Verification Automation

세 번째는 고려해야 할 사항은, 각 페이지에서 읽어온 목차체계가 표준으로 선언한 목차체계에 준하게 작성되었는지 검증하는 것이다. 검증은 두 가지로 구분되며, 목차상 숫자(예: 1.1.1)가 앞뒤 장표에서 순차적으로 작성되었는지, 그리고 목차상 텍스트는 중복되어 작성되었는지 여부를 검증하도록 개발하였다.

```
    dicA4ResultEnum = A4ResultList_.GetEnumerator();

    /*  Repetition as many entries as are stored in the array */
    while (dicA4ResultEnum.MoveNext())
    { // Extracting string from Chapter(e.g. "Ⅱ")
      strChapter = dicA4ResultEnum.Current.Key;
      lstRangeList = (List<string>)dicA4ResultEnum.Current.Value;
      iListCnt = lstRangeList.Count;  // lstRangeList.Count -> 14

    /* Repetition as many entries as are stored in the arrary */
      for (int i = 1; i < iListCnt; i++)
      {
       /*  lstRangeList[0].Split('^')
            Path/file name      -> [0]: e.g. "\\Ⅱ. OOO.pptx"
            Slide number        -> [1]: e.g. "1"
            Table of contents -> [2]: "1.1 OO"
            Table of contents level -> [3]: "2"
            Pattern type        -> [4]: "N.N"
            Item value          -> [5]: "1.1"          */
        aryRowData = (string[])lstRangeList[i].Split('^');
        sbContent.Remove(0, sbContent.Length); //Current table of
content
        sbContent.AppendFormat(@"{0}", aryRowData[2].ToString());

        iLevel = int.Parse(aryRowData[3]); // Current table of
content level
        sbPattern.Remove(0, sbPattern.Length);
        sbPattern.AppendFormat(@"{0}", aryRowData[4]); // Pattern
type
        // Extracting the number of decimal place from table of
content
        iFindPos = aryRowData[5].IndexOf('.');
         if (0 < iFindPos)
         {  // The integer part in the current item
             iInteger   =   int.Parse(aryRowData[5].Substring(0,
iFindPos));
             iResultData                                    =
int.Parse(aryRowData[5].Substring((iFindPos +
                 1)).Replace(@".", @""));  // the part of decimal point
         } else {    // The integer part in current item value
             iInteger = int.Parse(aryRowData[5]);
             iResultData = 0;
         }
         // Previous location data for comparison
         iFindPos = 0; // Initialize temporary variable
         iPrevIndex = ((0 == i) ? 0 : (i - 1));  // Previous location
         aryCompData = (string[])lstRangeList[iPrevIndex].Split('^');
         sbPrevContent.Remove(0, sbPrevContent.Length);
         sbPrevContent.AppendFormat(@"{0}",
aryCompData[2].ToString());
         iPrevLevel = int.Parse(aryCompData[3]); // table of content
level
         sbPrevPattern.Remove(0, sbPrevPattern.Length);
         sbPrevPattern.AppendFormat(@"{0}", aryCompData[4]);

         iFindPos = aryCompData[5].IndexOf('.');
         if (0 < iFindPos)
         {  // Decimal part in item value of previous page
            iPrevInteger = int.Parse(aryCompData[5].Substring(0,
iFindPos));
            iPrevResultData                                 =
int.Parse(aryCompData[5].Substring((iFindPos
                + 1)).Replace(@".", @""));  // Part under the decimal
point
         } else { // Integer part in item value of previous page
            iPrevInteger = int.Parse(aryCompData[5]);
            iPrevResultData = 0;
         }

      /* Assignment data to error list object after comparing values
*/

         // The case that table of contents level is same
         if (iLevel == iPrevLevel)
         { // The case that level type is first and table of content
is "N"
            if    ((1    ==    iLevel)    &&    (true    ==
aryRowData[4].Equals(@"N")))
            {
              if ((iPrevInteger + 1) != iInteger)
              { // Add to the error list object
                 lstErrorContents_.Add(lstRangeList[i].ToString());
              }
            } // The type of table of content level is "N.N"
            else if (true == aryRowData[4].Equals(@"N.N"))
            {
               if (iPrevInteger == iInteger)
               {
                  if ((iPrevResultData + 1) != iResultData)
                  {   // Function for type processing
                      ContentsTypeProcessing();
                  }
               }
               else if ((iPrevInteger + 1) != iInteger)
               {   // Add to the error list object
                  lstErrorContents_.Add(lstRangeList[i].ToString());
               }
            }
         }
         // The case that table of contents level is different
         else  // The type of table of content level is "N.N"
         {
            if (true == aryRowData[4].Equals(@"N.N"))
                {
                 if (iPrevInteger == iInteger)
                 {
                    if ((iPrevResultData + 1) != iResultData)
                    {  // Add to the error list object

lstErrorContents_.Add(lstRangeList[i].ToString());
                    }
                 }
                 else if ((iPrevInteger + 1) != iInteger)
                 { // Add to the error list object
                    lstErrorContents_.Add(lstRangeList[i].ToString());
                 }
            }
         }
    }
```

Fig. 8. Source Code(Order & Content Analysis Part) of Table of Contents Verification Automation

③ Text 표준적용 자동화

텍스트의 글꼴, 크기, 색 및 기타 효과(기울기, Bold, 이텔릭체 등)와 관련하여 표준에 벗어난 텍스트를 자동 추출한 후, 지정한 표준을 일괄적으로 자동 적용하도록 개발하였다. 시뮬레이션에서는 '맑은고딕', '24 Font' 그리고 'RGB(28, 28, 28)'을 표준으로 적용하였다. (하기 소스코드 중 *는 기타 효과 테스트)

```
    StringBuilder sb = new StringBuilder(); // Temporary variable for
string processing
    string strFontName = @"맑은 고딕";  // Standard font name
    Int iFontSize = 24;                  // Standard font size
    Color cFontColor = new Color();      // Standard font color

    cFontColor = Color.FromArgb(28, 28, 28);              // Dark
gray
    for (int i = 1; i <= iSlidesCount; i++)    // Repetition by number
of slide
    {
        for (int j = 1; j <= iShapeSetCount; j++) //Repetition by
number of shape
        {
          PPTShapeObject = PPTShapesObject[j]; // Assign shape
object

          sb.Remove(0, sb.Length);

sb.Append(PPTShapeObject.TextFrame.TextRange.Font.Name);
   // Extracting font name from specified shape

          if (!sb.ToString().Equals(strFontName)) // Non-standard
font
          {
            PPTShapeObject.TextFrame.TextRange.LanguageID =
                MsoLanguageID.msoLanguageIDKorean;

            PPTShapeObject.TextFrame.TextRange.Font.NameAscii
=
                strFontName;  // Setting standard font -
English/number

PPTShapeObject.TextFrame.TextRange.Font.NameFarEast =
                strFontName;  // Setting standard font -Asia/Korea

PPTShapeObject.TextFrame.TextRange.Font.NameOther =
                strFontName;  // Setting standard font - etc
            PPTShapeObject.TextFrame.TextRange.Font.Name =
                strFontName;  // Setting standard font - basic font

            PPTShapeObject.TextFrame.TextRange.Font.Size =
                iFontSize;      // Setting standard font size
            PPTShapeObject.TextFrame.TextRange.Font.Color.RGB
=
                Utility.BGR(cFontColor); // Setting standard font
color

            PPTShapeObject.TextFrame.TextRange.Font.Bold =
                MsoTriState.msoTrue; // Font style - bold
            PPTShapeObject.TextFrame.TextRange.Font.Italic =
                MsoTriState.msoTrue; // Font syle - Italic
            PPTShapeObject.TextFrame.TextRange.Font.Underline
=
                MsoTriState.msoTrue; // Underline style - single
line
          }
        }
    }
```

Fig. 9. Source code of Text Standard Application Automation

④ Table 표준적용 자동화

표는 컨텐츠의 내용에 따라 표현방식이 매우 다양하여 일괄적으로 하나의 표준을 적용하는 것은 현실적으로 불가능하다. 예를 들어, 상단부분에 타이틀이 1~3 Depth까지, 그리고 좌측부분에는 타이틀이 존재할 수도 그리고 존재하지 않을 수 있으며, 존재하는 경우 1~3 Depth까지 그 경우가 다양하다.

Table 표준적용 자동화 개발에서는, 가장 일반적인 경우인 상단 1 Depth영역만 타이틀 효과('굴림체', 12 Font, RGB(255, 255, 255))를 적용하였고, Table 內 각 셀의 상단, 하단 그리고 좌/우 선에 대한 굵기, 색, 실선 유형을 일괄적으로 표준을 자동 적용하였다. 또한 표 내용부분 각 셀의 색 및 텍스트의 폰트를 자동 적용('돋음체', 10 Font, RGB(88, 88, 88))하였다.

```
    string strFontName_01 = @"굴림체";   // Font in title of table
    string strFontName_02 = @"돋음체";   // Font in content of table
    int iFontSize_01 = 12;   // Font size in title of table
    int iFontSize_02 = 10;   // Font seize in content of table

    Color cPenColor = new Color();  // Table border line color
    Color cContentsBGColor_01 = new Color(); //Background color
in first row
    Color cContentsBGColor_02 = new Color(); //Background color
in second row
    Color cContentsBGColor_03 = new Color(); //Background color
in third row
    Color cFontColor_01 = new Color(); // Font color in title of table
    Color cFontColor_02 = new Color(); // Font color in content of
table
    int iRowCnt = 0;        // The number of table row
    int iColumnsCnt = 0;  // The number of table column

    cPenColor = Color.FromArgb(72, 72, 72);//Color of table border
line
    cContentsBGColor_01 = Color.FromArgb(172, 172, 172); //
Background color
    cContentsBGColor_02 = Color.FromArgb(255, 255, 255);
    // Background color of odd line of table
    cContentsBGColor_03 = Color.FromArgb(242, 242, 242);// Even
line of table
    cFontColor_01 = Color.FromArgb(255, 255, 255);//Font color of
title
    cFontColor_02 = Color.FromArgb(88, 88, 88); // Font color of
content
```

Fig. 10. Source code(Variable Declaration Part) of Table Standard Application Automation

타이틀 영역과 컨텐츠 영역의 폰트, 크기, 색 및 각 셀의 배경색 그리고 선 색과 두께를 설정한 후, 문서내 Table Shape를 찾아 두 for 제어문(각 셀의 행/열 이동을 위해)을 활용하여 표준을 자동 적용하도록 구성하였다.

```
  foreach (PowerPoint.Slide slide in this.PPTPresentaion.Slides)
// Repetition by number of slide
  {  // Repetition by number of Shape
    foreach (PowerPoint.Shape shape in slide.Shapes)
    {
     PPTTable = shape.Table; // Selecting object of Shape
     iRowCnt = PPTTable.Rows.Count;
     iColumnsCnt = PPTTable.Columns.Count;

     for (int iR = 1; iR <= iRowCnt; iR++)  // Row change
     {
       for (int iC = 1; iC <= iColumnsCnt; iC++)//Column change
       {
         PPTCell = PPTTable.Cell(iR, iC);  // Current cell of table
         PowerPoint.LineFormat leftBorder =
```

```
PPTCell.Borders[PowerPoint.PpBorderType.ppBorderLeft];
            PowerPoint.LineFormat topBorder =

PPTCell.Borders[PowerPoint.PpBorderType.ppBorderTop];
            PowerPoint.LineFormat rightBorder =

PPTCell.Borders[PowerPoint.PpBorderType.ppBorderRight];
            PowerPoint.LineFormat bottomBorder =

PPTCell.Borders[PowerPoint.PpBorderType.ppBorderBottom];

            switch(iR)
            {
               case 1: // The area of title of table
               { // Setting the color and thickness of border line
                bottomBorder.ForeColor.RGB =
                     ColorTranslator.ToOle(cPenColor);
                bottomBorder.Weight = 2.25f;
                bottomBorder.Style = MsoLineStyle.msoLineSingle;
                leftBorder.ForeColor.RGB =
                      ColorTranslator.ToOle(cPenColor);
                leftBorder.Style = MsoLineStyle.msoLineSingle;
                rightBorder.ForeColor.RGB
=ColorTranslator.ToOle(cPenColor);
                rightBorder.Style = MsoLineStyle.msoLineSingle;
               // Setting the font of cell
               PPTCell.Shape.Fill.ForeColor.RGB =
                    ColorTranslator.ToOle(cContentsBGColor_01);
               strFontName = strFontName_01;
               iFontSize = iFontSize_01;
               cFontColor = cFontColor_01;
               break;
              }
              default:
              {
               // Setting the color and thickness of border line
               bottomBorder.ForeColor.RGB =
                     ColorTranslator.ToOle(cPenColor);
               bottomBorder.Style = MsoLineStyle.msoLineSingle;
               leftBorder.ForeColor.RGB =
                        ColorTranslator.ToOle(cPenColor);
               leftBorder.Style = MsoLineStyle.msoLineSingle;
               rightBorder.ForeColor.RGB =
                         ColorTranslator.ToOle(cPenColor);
               rightBorder.Style = MsoLineStyle.msoLineSingle;
               // Setting the font of cell
               if (0 == (iR % 2))
               {
                  PPTCell.Shape.Fill.ForeColor.RGB =
                       ColorTranslator.ToOle(cContentsBGColor_02);
                }
                else
                {
                  PPTCell.Shape.Fill.ForeColor.RGB =
                       ColorTranslator.ToOle(cContentsBGColor_03);
                 }
                 strFontName = strFontName_02;
                 iFontSize = iFontSize_02;
                 cFontColor = cFontColor_02;
                  break;
               }
            }
            PPTCell.Shape.TextFrame.TextRange.Font.Size =
            iFontSize;          // Font size

            PPTCell = null;
          }
        }
      }
    }
```

⑤ 선/화살표 표준적용 자동화

도식화된 장표 內 선과 화살표에 대해 표준을 일괄적으로 자동 적용하는 것으로, 선의 색, 굵기 및 유형(실선 or 점선 등) 그리고 선의 시작점과 끝지점의 모양(원형, 화살표 등) 그리고 화살표 머리 부분의 유형(꺽쇠 스타일 or 삼각형 스타일 등)이 그 대상이 된다.

선/화살표 표준적용 자동화 개발에서는, 선 굵기는 1.5f, 색은 RGB(127,127,127), 시작부분은 원형 그리고 끝 부분은 삼각형 스타일의 화살표를 표준으로 하였다.

```
    PPTApp = new PowerPoint.Application();
    PTApp.Visible = MsoTriState.msoTrue;

    // Assign PowerPoint object to PowerPoint Documents
    PPTPresentations = PPTApp.Presentations;

    // Open the PowerPoint document
    PPTPresentations.Open(pptOpenInfo.FullPath,
MsoTriState.msoFalse, MsoTriState.msoFalse,
MsoTriState.msoTrue);
    // Total count of presentation
    iPresentationsCount = PPTPresentations.Count;
    PPTPresentaion = PPTPresentations[iPresentationsCount];

    foreach (PowerPoint.Slide slide in this.PPTPresentaion.Slides)
 // Repetition by number of slide
    {
      // Repetition by number of shape
      foreach (PowerPoint.Shape shape in slide.Shapes)
      {
         // Checking whether shape is line
         if (MsoShapeType.msoAutoShape != shape.Type)
         {
            continue;
         }
         shape.Line.Weight = 1.5f;  // Setting the thickness of line
         shape.Line.ForeColor.RGB =
            ColorTranslator.ToOle(Color.FromArgb(127, 127, 127));
       // Setting the color of line
         shape.Line.BeginArrowheadStyle
=MsoArrowheadStyle.msoArrowheadOval;
         //  Setting the beginning of line as a cycle
         shape.Line.EndArrowheadStyle =
              MsoArrowheadStyle.msoArrowheadTriangle;
        // Setting the end of line as a arrow
       }
    }
```

Fig. 12. Source code of Line/Arrow Standard Application Automation

4.4 시뮬레이션 결과

시뮬레이션은 공공사업의 제안서를 기준(세로형 제안서, 300페이지 이내)으로 수행하였으며, Human의 Execution Time은 실제 품질활동을 수행하는 구성원의 행동을 관찰을 한 결과이며, Robot의 Execution Time은 동일한 작업을 프로그램이 수행한 시간을 의미한다. Human과 Robot을 활용한 문서품질활동의 수행결과는 다음과 같다.

① 오탈자 검증 자동화

■ 시뮬레이션 결과

Table 4. The Result of Simulation

Automation Area	Simulation Results					
	Human		Robot		Productivity	
	Execution Time	Error Case	Execution Time	Error Case	Execution Time	Error Rate Improvement
①Typographical Error Verification	60.0 min	27	2.1 min	3	96.5% ↑	88.9%↑
②Table of Contents Verification	35.0 min	0	0.7 min	0	98.0% ↑	100%↑
③Text Standard Application	1,650.0 min	127	1,110.0 min	0	32.7% ↑	100%↑
④Table Standard Application	182.0 min	7	119.6 min	0	34.3% ↑	100%↑
⑤Line/Arrow standard application	635.0 min	16	457.2 min	0	28.0% ↑	100%↑

오타를 수정하는데 있어서 사람마다 다소 차이는 있으나, 평균적으로 한 명이 약 15분이 소요되며, 일반적으로 하나의 제안서에 2~3명이 2회 수행하는 것으로 관찰됨에 따라, 최종 제안서를 제출하기까지 평균적으로 총 60분이 소요되는 것으로 예상된다. 동일한 제안서를 대상으로 Robot을 활용하여 오탈자를 점검한 결과, 2분 6초가 소요되어 96.5%의 생산성이 향상됨을 확인할 수 있었다. 특히, 오탈자 탐지에 대한 오류는 사람이 수행하였을 때보다 88.9%가 향상되었다. 다만, Robot의 오류 3건은 하나의 Input Box에 "Enter Key"를 사용하여 문장을 작성함으로써 다른 단어/용어로 인식된 결과로 프로그램상 오류는 아닌 것으로 파악되었다.

▪RPA 솔루션개발 시 고려사항

MS-Office에 등록된 표준어 Dictionary를 기반으로 오탈자를 검색함으로써, 정확한 오탈자 검색에 한계가 있는 것도 사실이다. 예를 들어, BigData, BlockChain, IoT 등은 오탈자가 아닌 새롭게 등장한 단어임에도 표준어 Dictionary에 등록되지 않아 오탈자로 검출되는 대표적인 사례라 할 수 있다. 따라서 이러한 한계를 극복하기 위해 MS-Office Dictionary 외 개별적인 데이터베이스를 구성하여 더블체크를 한다면 정확도를 보다 높을 수 있을 것으로 예상된다.

② 목차 검증 자동화

▪시뮬레이션 결과

목차 오류여부를 사람이 수행하는 경우, 300페이지의 제안서를 점검하는데 일반적으로 1명이 약 35분 소요되는 것으로 관찰되었다. 동일한 제안서를 Robot으로 검사 시에는 42초로 단축되어 생산성이 98.0% 향상됨을 확인하였다. 다만, 두 가지 모두 잘못된 오류점검 결과는 없었다.

▪RPA 솔루션개발 시 고려사항

시뮬레이션에서는 가장 일반적인 형태의 한 가지 목차 체계(1 > 1.1 > 1.1.1 > 가 > 1) > 가))만 실험을 하였으며, 상용 솔루션에서는 활용도 제고를 위해 다양한 목차 체계를 적용할 수 있도록 개선하는 것이 필요하다. 또한 오류검사 결과를 시각적으로 표현하기 위해 엑셀로 출력하고 오류가 예상되는 부분을 색(예: 붉은색)으로 구분할 경우 보다 효율적일 것으로 예상된다. 이를 위해서는 트리구조 객체인 TreeNode 및 TreeView를 활용해야 보다 쉽게 컨트롤이 가능할 것으로 사료된다.

③ Text 표준적용 자동화

▪시뮬레이션 결과

텍스트 표준을 적용하는 것은, 일반적으로 제안팀 구성원이 아닌, 편집디자인을 담당하는 디자이너가 수행한다. 시뮬레이션의 대상인 공공사업 제안서의 경우 구성도가 많아 편집 디자이너 4명이 5일 동안 작업하는 것이 일반적이다.

텍스트 표준 적용 시, 편집 디자이너의 숙련도에 따라 다소 차이는 있으나, 한 페이지당 평균적으로 5분 30초가 소요되는 것으로 관찰되었으며, 300 페이지를 적용했을 때 약 1,650분(27.5시간)이 소요되는 것으로 관찰되었다. 이것을 Robot이 수행했을 경우, 한 페이지에 텍스트의 양에 따라 소요시간에도 차이는 있으나, 평균적으로 3분 42초가 소요되어 32.7%의 업무생산성이 향상되는 것으로 확인되었다.

이전 시뮬레이션 결과와 다르게 생산성 향상이 상대적으로 낮은 이유는, 오피스 프로그램 자체가 UI가 실행된 상태에서 COM을 이용하여 파일을 조작 및 제어할 수 있도록 구성되어 있기 때문이다. 시뮬레이션 대상이 되는 제안서의 경우, 적은 단어로 된 많은 텍스트 Shape를 for 제어문을 통해 체크하고 수정해야 하는데, UI가 실

행된 상태에서 작동함으로 이에 따라 부하가 발생하며, 또한, 자동화 대상이 되는 파일이 용량이 크고, 컴퓨터 사양이 낮은 것도 그 원인이라 할 수 있다.

■ RPA 솔루션개발 시 고려사항

현재의 소스 코드는 제안서 본문영역에 존재하는 텍스트의 서체, 크기, 색, 그리고 효과(기울기/Bold/이텔릭체 등)에 대해 표준을 일괄적으로 적용하는 형태이나, 다양성을 줄 경우 보다 유용하게 사용할 수 있을 것으로 사료된다. 예를 들어, 텍스트의 크기가 6~9인 경우는 일괄적으로 8로, 10에서 12인 경우에는 11로 적용하는 것이다. 왜냐하면, 한 페이지 안아서 컨텐츠의 양에 따라 텍스트의 크기도 다소 달라지기 때문이다.

또한, 문서의 Header 영역에는 목차 및 주제문장(Governing Message)이 위치하는데, 이 영역의 텍스트는 본문영역과 다르게 텍스트의 폰트, 크기, 색 뿐만 아니라 위치까지도 정해져 있어서, 이를 자동화할 경우 작업 생산성에 크게 도움이 될 것으로 예상된다.

④ Table 표준적용 자동화

■ 시뮬레이션 결과

Table 표준을 적용하는 것은 텍스트 표준적용과 마찬가지로 제안팀 구성원이 아닌, 편집디자인을 담당하는 디자이너가 수행한다.

시뮬레이션의 대상이 된 제안서의 경우, 300페이지 중 Table이 들어간 페이지는 52페이지에 해당되었으며, 해당 컨텐츠의 내용에 따라 Table의 크기와 유형이 매우 다양하였다. 편집 디자이너의 숙련도에 따라 다소 차이는 있으나, 한 페이지당 평균적으로 3분 30초가 소요되는 것으로 관찰되었으며, 제안서 전체를 디자인 했을 때 182분이 소요되었다. 이것을 Robot이 수행했을 경우, 한 Table을 디자인하는데 평균적으로 2분 18초, 전체적으로는 119분 36초가 소요되어 34.3%의 업무생산성이 향상되는 것으로 확인되었다.

낮은 처리속도는 '③ Text 표준적용 자동화'와 동일한 원인으로, 마찬가지로 오피스 프로그램 제어를 위한 COM 개체를 사용하기 때문이다.

■ RPA 솔루션개발 시 고려사항

Table에 대한 자동화를 적용하는데 가장 큰 문제점은, 앞에서 언급했듯이 컨텐츠의 내용에 따라 타이틀 영역이 매우 다양한 형태를 보이고 있으며 획일적으로 하나의 표준을 정하는 것은 불가능하다는 것이다. 따라서 제안서에 사용되는 표의 유형을 정의하고 이에 따른 자동화 기준을 수립하는 것이 현실적인 방안이라 생각된다. 만약, 이러한 표의 유형을 확정하기 어렵다면, 각 셀에 대한 배경색/선굵기, 셀 안의 텍스트에 대한 폰트/크기/색만 자동화를 적용하고, 타이틀 부분에 대한 강조(배경색 및 텍스트 Bold 처리)는 수작업으로 진행하는 것이 현실적인 방법이다.

⑤ 선/화살표 표준적용 자동화

■ 시뮬레이션 결과

선 그리고 화살표의 표준을 적용하는 것은 텍스트와 Table 사례와 마찬가지로 제안팀 구성원이 아닌, 편집디자인을 담당하는 디자이너가 수행한다.

제안서를 작성하는데 있어서 선의 굵기, 색 그리고 화살표 머리모양이, 실제로 개인마다 조금씩 다른 기준을 적용하였고, 그리고 참고용으로 활용한 제안서가 매우 다양하여, 디자인적으로 표준이 없는 대표적인 요소에 해당된다.

시뮬레이션의 대상이 된 제안서의 경우, 300페이지 중 127페이지에 선/화살표가 작성되었다. 시뮬레이션은 선/화살표를 하나라도 들어간 페이지 단위로 하였다. 편집 디자이너가 직접 작업한 것은, 그 숙련도에 따라 다소 차이는 있으나, 평균적으로 한 페이지당 5분이 소요되는 것으로 관찰되었으며, 제안서 전체를 디자인 했을 때 635분이 소요될 것으로 예상되었다(편집 디자인 작업시 선/화살표만 모아서 작업하는 방식이 아님으로 예상치로 표현함). 이것을 Robot이 수행했을 경우, 한 Table을 디자인하는데 평균적으로 3분 36초, 전체적으로는 457분 12초가 소요되어 28.0%의 업무생산성이 향상되는 것으로 확인되었다. 사람이 편집 디자인을 수행한 경우 오류가 다수 발견되었는데, 이는 작업의 어려움보다는 장시간 단순 반복적 작업수행에 따른 단순한 휴먼에러(Human Error)이다.

낮은 처리속도는 앞 사례와 동일하게 오피스 프로그램 제어를 위한 COM 개체를 사용하기 때문이다.

■ RPA 솔루션개발 시 고려사항

선의 종류 중 실선과 점선이 의미하는 것이 상이하여(예: 실선은 On-Line, 점선은 Off-Line) 일괄적으로 적용하는 것은 오히려 또 다른 불필요한 작업을 야기시킬 것으로 예상된다.

RPA 솔루션 개발 시 추가적으로 고려할 사항은, 컨텐츠의 내용에 따라 선과 화살표의 크기 및 굵기가 상이한 경우가 있으므로, 일괄적 자동화보다는 선의 굵기 범위별로(예: ③ Text 표준적용 자동화 예시와 동일) 표준을 적

용하는 것이 업무생산성 향상에 좀 더 도움이 될 것으로 예상된다.

5. 결론

시뮬레이션 결과, 문서 품질점검 자동화에서는 업무생산성이 평균적으로 97.3%, 편집 디자인 자동화에서는 평균적으로 31.7% 향상된 것으로 분석되었다. 물론 이러한 측정 결과가, 비즈니스 문서에서 자동화가 가능한 모든 영역을 대상으로 수행한 것은 아니며, RPA의 기본 관점인 '단순 반복적인 작업의 자동화 및 휴먼에러 제거'라는 측면에서 Robot적용이 가능할 것으로 예상되는 일부 영역만을 그 대상으로 수행한 결과이다. 그럼에도 불구하고, 우리가 그 유용성에 막연함을 갖았던, 그리고 분명 필요하지만 그 적용가능성에 의문이 있었던 문서의 품질점검 및 편집 디자인 부분에 Robot 적용의 가능성을 확인한 것에 큰 성과가 있었다고 생각한다.

편집 디자인 관련된 자동화 영역인 '③Text 표준적용 자동화', '④Table 표준적용 자동화' 그리고 '⑤선/화살표 표준적용 자동화'의 생산성 향상이 상대적으로 낮은 것이 실제현장에 적용할 수 없다는 것을 의미하지는 않는다고 생각한다. 예를 들어, 퇴근시간에 Robot을 실행시키고, 다음날 출근하여 결과물을 받아 품질을 향상시키는 작업, 즉 Value있는 업무에 집중한다면, RPA의 목적에 충분히 부합될 수 있다고 예상된다.

물론, 앞에서 살펴본 자동화 방안을 실제 비즈니스 환경에 적용하기 위해서는 기능적으로 보완할 부분이 많이 존재한다. 앞에서 기술한 'RPA 솔루션개발 시 고려사항' 외에도 보다 범용적으로 활용하기 위해, 다양한 템플릿을 적용(예: 텍스트, Table, 화살표 등 다양한 표준을 정의한 문서)할 수 있도록 개발해야 한다. 이를 위해 해당 표준을 지정하는 별도의 입력폼(Form) 및 이력관리가 필요하며, 이는 반드시 시스템상의 입력창(Input Box)을 의미한다기 보다는, 표준을 정의하는 워드 파일 혹은 엑셀 파일로도 관리가 가능할 것으로 예상된다.

또한, 이러한 문서 자동화와 관련된 Robot은, 출판 분야로 확대 적용이 가능하다. 학습서, 교과서, 백서 등과 같은 문서는 매우 정형화되고 패턴화된 문서양식으로, 해당 작업에서 활용되는 디자인 도구(예: Adobe Indesign 및 Adobe Illustrator 등)를 컨트롤하는 자동화 Robot을 개발한다면 업무생산성 향상에 매우 큰 영향을 미칠 것으로 예상된다. 또한, 공공기관의 정형화된 보고서의 경우 아래아 한글로 작성을 하는데, 이 또한 아래아 한글 SDK를 활용한다면, 문서 품질점검과 관련하여 업무생산성이 혁신적으로 향상될 것으로 예상된다.

무엇보다 중요한 것은, 이러한 생산성향상이 곧 인간노동(Human Labor)을 디지털노동(Digital Labor)으로 대체할 수 있다는 방향으로 해석되는 것을 경계하고 싶다[15,16]. 오히려, 인간노동(Human Labor)를 단순 반복적인 업무에서 벗어나 Robot이 할 수 없는, 즉 인간의 사고와 인지적 판단이 필요한 영역에 집중함으로써 업무의 Value를 높이는 활동에 집중할 수 있는 출발점이 되어야, RPA의 근본적 목적에 부합되리라 생각한다.

REFERENCES

[1] G. S. Kim, G. Y. Lee & M. J. JO, (2017). *RPA implement and Service innovation (Focusing on financial industry cases).* Seoul : Samjong KPMG

[2] D Schatsky, C Muraskin & K Iyengar. (2017). *Robotic process automation, A path to the cognitive enterprise.* New York : Deloitte Consulting.

[3] K. Y. Lee. (2018). *The simple repetition task is entrusted to the robot, but focuses on Work Smart rather than Work Less.* Dong-A Business Review(DBR). http://dbr.dong.com

[4] S. M. Ahn. (2018) *Chat-Bot. Accelerates introduction of RPA such as robot advisor.* METRO http://www.metroseoul.co.kr

[5] E. Y. Lim. (2018) *Software robots simulating human tasks,* Magazine of JoongangilboPlus https://jmagazine.joins.com

[6] J. H. Jung. (2017). *The Fourth Industrial Revolution Winds at Office(The era of Robots in Biz. operation).* Seoul : POSCO Economic Research Institute

[7] Y. D. Kim.. (2013). *Requirement Tracing Method Using Matrix in Agile Development Process.* Doctor's Thesis, Junnam University Graduate School, Gwangju.

[8] J. H. Park. (2017). *White-Color worker's tool to improve productivity*, Seoul : Institute for information & communication technology promotion

[9] C. Le Clair. (2017). *Robotic Process Automation, Q1 2017(The 12 Providers That Matter Most And How They Stack Up)*, Cambridge : The Forrester Wave

[10] H. J. Cho1 & K. M. Jeong. (2019). A Study on the Connective Validity of Technology Maturity and Industry for Core Technologies based on 4th Industrial Revolution. *Journal of the Korea Convergence Society 10(3), 49-57. 10.15207/JKCS.2019.10.3.049*

[11] H. S. Ryou & D. H. Lee. (2016). A Comparative Study on Analytical Tools of Business Model. *Journal of Digital*

Convergence, 14(5), 137-147. 10.14400/JDC.2016.14.5.137

[12] S. W. Kang. (2018) *A Study on the Performance and Success Factors of Exporting Company's Office Automation Solutions(Based on RPA Application Cases).* Korea Trade Research Association

[13] J. Y. Kim & W. G. Heo. (2018). A Study on Issues and Tasks of Humanity and Social Science in a Fourth Industrial Revolution Era. *Journal of Digital Convergence, 16(11), 137-147.*
DOI: 10.14400/JDC2018.16.11.137

[14] H. M. Kim. (2013). *A Case Study of Android Applications Development with Agile Methodology.* Doctoral dissertation, Soongsil University, Seoul.

[15] J. H. Oh, (2019) *In the employment era without employers, We need a social safety net.* Gyeonggi Research Institute

[16] S. H. Ahn & M. H. Lee, (2016) *Fourth Industrial Revolution Impact : How it Changes Jobs.* Korean Academic Society Of Business Administration

현 영 근(Young-Geun Hyun) [정회원]

· 2018년 8월 ~ 현재 : 아주대학교 공과대학 산업공학과 석박사통합과정
· 2017년 8월 ~ 2018년 7월 : 아주대학교 공과대학 산업공학과 석사과정
· 2018년 1월 ~ 현재 : SK 주식회사 C&C DT 전략 Marketing 그룹
· 2004년 12월 ~ 2017년 12월 : SK 주식회사 C&C 제안전략 Consultant
· 1999년 8월 ~ 2004년 11월 : SI Computer programer
· 관심분야 : 융합기술연구, AI, Business Automation
· 저서 : 고객의 마음을 움직이는 제안전략
· E-Mail : hyunyg@ajou.ac.kr

이 주 연(Joo-Yeoun Lee) [종신회원]

· 2002년 2월 : 인하대학교 대학원경영학박사
· 2014년 9월 ~ 현재 : 아주대학교 공과대학 산업공학과 교수
· 2015년 2월 ~ 2018년 1월 : 산업통상자원부 산업융합촉진 국가옴부즈만(차관급)
· 2016년 7월 ~ 2019년 6월 : 한국빅데이터서비스학회 학회장
· 2007년 7월 ~ 2011년 6월 : 한국산업정보학회 회장
· 2011년 12월 ~ 2014년 3월 : POSCO ICT 그린사업부문장(전무)
· 2005년 2월 ~ 2011년 11월 : SK C&C 전략마케팅본부장(상무)
· 1999년 12월 ~ 2005년 1월 : Oracle 전략솔루션실장(Director)
· 관심분야 : 산업융합기술, 비즈니스인텔리전스, 서비타이제이션, 스마트그리드
· E-Mail : jooyeoun325@ajou.ac.kr