**B9IS105 - Enterprise Information Systems**



***Evaluation of order management business process in Coca Cola***

Module Lecturer: Dr. Obinna Izima

Akhil Ravuri – 10610852

Sharon Farrell - 10581193

Shastri Telikicherla – 10606539

**Contents**

1. Glossary ………………………………………………………………………………………………. 3
2. List of illustrations ………………………………………………………………………………. 4
3. About Coca-Cola …………………………………………………………………………………. 5
4. Introduction

* Scope of the document ……………………………………………………………………. 5
* Limitations ………………………………………………………………………………………. 5

1. Enterprise Information Systems & their Architecture …………………………… 6
2. Coca Cola’s legacy order management process

* Brief explanation ………………………………………………………………………………. 8
* Challenges ………………………………………………………………………………………….9

1. Improvised/New order management process

* Technology & Tools …………………………………………………………………………. 10
* Architecture ……………………………………………………………………………………… 11
* Business Improvements ……………………………………………………………………. 12
* Advantages and Disadvantages ………………………………………………………… 13

1. Recommendations ………………………………………………………………………………. 14
2. References ……………………………………………………………………………………………. 15

**Glossary**

1. EIS – Enterprise Information Systems
2. IT – Information Technology
3. NFR – Non Functional Requirements
4. COM – Component Object Model
5. ERP – Enterprise Resource Planning
6. SOA – Service Oriented Architecture
7. MAS – Multi Agent System
8. SOAP – Simple Object Access Protocol
9. WSDL – Web Service Description Language
10. RAS – Remote Access Service
11. RPA – Robotic Process Automation
12. SCM – Software Change Management
13. CRM – Customer Relationship Management

**List of Illustrations**

1. Figure 1 - Framework for scenario-based EIS software architecture analysis
2. Figure 2: Database-centred layered architecture
3. Figure 3: Web services architecture
4. Figure 4: BXD's SCM software architecture alternatives
5. Figure 5: Manual order management process before implementing RPA
6. Figure 6: New automated process of order management system using Microsoft Power Platform

**About Coca Cola**

Coca-Cola bottling company also known as Coca-Cola united is one of the largest and privately held Coca-Cola bottlers in United States and its parent company is famously known as Coca-Cola, it is not just a brand but a household name, there might be hardly an individual who has not heard about Coca-Cola. It was founded in 1902 and has its headquarters in Birmingham, Alabama (United States). It serves over 200,000 customers and uses thousands of trucks to deliver more than 600 product lines ranging from small stores to super stores and mega retailers.

**Introduction**

We referred to Microsoft’s customer success story whitepaper on Coca-Cola bottling company (<https://customers.microsoft.com/en-us/story/845187-coca-cola-bottling-company-united-consumer-goods-power-automate>) and extensively studied their order management business process and understood how they used technology and automation to improvise the process. Based on our understanding, we have prepared this report which outlines their legacy order management systems, the steps they took in terms of technology to improvise it and what are our recommendations that can further improve or add value to the business process.

***Scope***

* The scope of this report is only limited to business process study of order management system of Coca-Cola bottling company/ Coca-Cola United.
* Based on our understanding of the innovation/new architecture design, we have discussed and provided our recommendations at the end of this report which are solely our understanding and might be debatable or require discussion before any actual implementations.

***Limitations***

* Our understanding on Coca-Cola’s order management business process and the technical innovations done by them are solely limited and are as described in the Microsoft’s whitepaper and other technical articles, whose links are provided under references.

**Enterprise Information Systems & their Architecture**

Information Technology (IT) plays a dominant role in today's industrial automation. For example, in designing a controlled drive system, an engineer working in the 1980s would deal primarily with mechanical and electronic components, whereas 90% of today's engineering time is devoted to the tasks on information systems [1]. Enterprise Information Systems (EISs) are the key IT assets for industrial enterprises to organize, plan, schedule, and control their business processes [2]. In particular, for supply chain management, EISs have become critical enablers for modern enterprises to streamline processes and achieve effectiveness, efficiency, competency, and competitiveness of the material flow. An essential component of an EIS is software architecture. Software architecture describes a set of system components as well as their topological relations in an EIS [3]. An advantage of architecture analysis lies in the early decisions about a software system's high level design [4]. Due to the importance of software architecture, the study in this field is emerging.

Researchers have recently proposed many software architecture descriptions to accelerate industrial applications [5]. These descriptions can be classified into domain-specific EISs [6], distributed real-time control [7], [8], embedded and dependable systems [9], [10], agent platforms [11], [12], and service-oriented architecture [13], [14]. In designing and implementing an EIS, software architecture must support the key business drivers; these drivers are also referred to as quality attributes or non-functional requirements (NFRs). Such a support is aligned with the enterprise missions and adds value to the system level [3]. As a proof-of-concept, existing architecture only supports single NFR, e.g., extensibility [8], fault tolerance [9] and so on.

When selecting software architecture for an EIS, industrial engineers need to consider multiple and often conflict NFRs. For example, a system's flexibility and real-time performance are conflicted with each other and must be balanced in software development [12]. Despite the increasing number of the proposals of software architecture options, fewer methods are available to evaluate software architecture against the requirements of a specific application. This has caused a hurdle in developing an EIS since the objectives of software architecture must be simultaneously considered to meet the requirements of today's IT-driven industrial automation. To select software architecture, a user-oriented method has been proposed to evaluate software architecture choices [15]; key NFRs are reviewed and the quality attribute scenarios are leveraged to assess the degree to which software architecture choices have influenced the fulfillment of the NFRs.

### Types of Software Architecture

Software architecture describes system components as well as their external properties, and the internal relations of components [3]. This research field emerged in the 1990s when the major work was to establish the fundamentals of software architecture including description languages, formal logic, architectural styles, design patterns, and the like. In the context of industrial informatics, software architecture represents the business structures and processes of an EIS, and it is a vital tool to support the assessment of design operations at an early stage [2].

A significant contribution to the development of software architecture is the patterns codification which can be used as the blueprint of components, constraints, and their relations. Patterns define the general solutions that can be reused to accelerate the software development process. Some methods to apply the operational patterns for the EIS design are as follows.

General purpose software packages encapsulate data structures and algorithms to implement a generic but customizable solution of business problems based on the best practices. Market-leading providers include SAP AG, Oracle Corporation, and Baan Co. In these packages, many operational patterns are exploited: database-centered data sharing, pipeline-based data processing, event driven message invocation, to name a few. The packaged software tools have been adopted by a variety of enterprises to optimize their business processes [2].

Domain-specific software architecture is tailored to EISs in a specified domain. Such architecture includes some special components which differ from common components of generic software architecture. For example, an industry-oriented ERP is capable of accommodating the requirements especially for a certain industry domain; some insignificant software elements and tools included in generic software package can be removed to reduce the complexity [6]. Major enabling technologies for DSSA domain-specific software architecture include Enterprise Java Beans (EJB), Microsoft's Component Object Model (COM+), and business component factory [20].

Distributed computing involves several interacting elements coordinated to achieve a system-level goal. Distributed programming typically falls into one of the following architectural options: client-server, n-tier architecture, and peer-to-peer. For the application of distributed computing, Ferrolho and Crisóstomo [7] presented distributed architecture to develop a flexible manufacturing cell using the Ethernet network. In addition, programming languages with parallel and concurrency supports (e.g., C++) and middleware technologies (e.g., CORBA) are among the key enablers for distributed computing.

Agent and multi-agent systems (MAS) have received much attention recently and they have been deployed widely. An agent is an autonomous entity situated in the environment; whereas a MAS is composed of a group of agents; the agents within a MAS can cooperate or compete each other to achieve the goals at the system level [21]. MASs have been successfully applied in manufacturing [11], behavior scheduling [12], workflow management [22], and business rules integration [23]. For example, Metzger and Polaków have investigated the applications of MSA in the process control [19].

Service-oriented architecture (SOA) can be viewed as a recent advance in integrating heterogeneous platforms including legacy software tools [2]. A SOA allows an EIS to extend its capabilities by applying reusable software modules so that the development cost can be reduced without reinventing a wheel [24]. Equipped with methods like Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Discovery, Description, and Integration (UDDI), the SOA has been introduced as a critical enabling technology to EISs [13], [14].

### Scenario-Based Software Architecture Analysis

The user-oriented method is proposed to facilitate the decision-makings related to EIS software architecture. The evaluation in the proposed method, according to Section II-A, shall be performed based on the intended business and quality attribute goals. This is because the way software architecture supports the driving NFRs determines how an EIS will behave. The performance of an EIS in turn will shape the business strategies and technical capabilities of enterprises.

The benefits of fulfilling the EIS NFRs are unarguable [10]; however, there are many unsolved practical issues when NFRs are considered in the implementation of EISs. For example, NFRs are usually subjective and hard to be quantified. This calls for qualitative methods to reason how well the EIS software can meet the NFRs [18]. Moreover, users express their missions with different terminologies [26], even though there are several standards related to NFRs, e.g., the International Organization for Standardization and International Electro-technical Commission (ISO/IEC) 25030. Terminological interferences relevant to the EISs are unavoidable, as illustrated in the first and second columns of Table I. Another challenge is that an EIS has to balance a set of the conflict objectives to determine its software architecture. Some examples of the conflict objectives are flexibility vs. productivity, scalability vs. reliability; moreover, all of the objectives contribute to the cost factor. It is necessary for industrial information engineers to consider all of the objectives simultaneously at the system level.

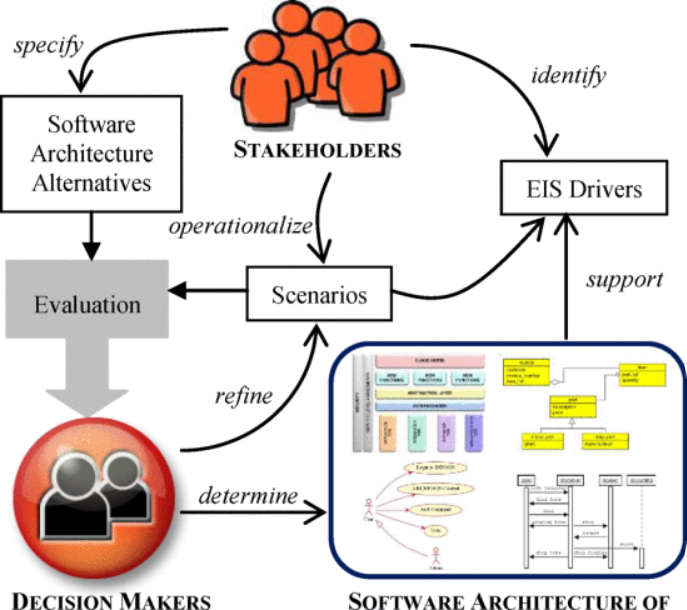


Figure 1: Framework for scenario-based EIS software architecture analysis

To meet practical challenges in evaluating software architecture based on the given NFRs, we propose a scenario-based method as shown in Fig. 1, where boxes and arrows represent entities and activities respectively. The core component—“evaluation”—is highlighted in a shaded callout. As depicted in Fig. 1, the scenarios play two important roles in the evaluation: firstly, they allow the abstract NFRs to be concretely defined, operationally measured, and meaningfully communicated among the stakeholders; secondly, they link architecture choices to the satisfaction of the EIS drivers, which helps the management to make an informed decision about the system that is best suited to their needs. The rest of this section describes key activities of the proposed method in details.

### A. Identifying NFRs

Without losing the generality, an EIS for an academic department is used as an example in this section; it is called a research administration system (RAS) in the rest of paper. The purpose of the RAS is to manage the research expenses for a group of researchers. The participants in the RAS can be easily extended to multi-institutions even multi-countries. In the proposed RAS, participators are able to register and take part in project events, upload or download project reports, and claim costs for the tasks. The project activities are planned and the progress is updated periodically.

RAS is a typical EIS, which is capable of maintaining data and providing the interfaces to users to access, transfer, process, and report data related to the research projects. Meanwhile, the fulfillment of several NFRs exhibited by the system will determine the success of RAS. Despite that many NFRs have been listed and discussed in Table I; when a specific application of EIS is considered, a sub-set of NFRs can be selected based on the priorities of the goal in the application. In other words, a NFR might be crucial to one application but insignificant to another application. While all of the NFRs should be considered, an industrial information engineer should select a few of major NFRs in applying the proposed method to simplify the evaluation. The following NFRs are considered as major NFRs in the example EIS.

Performance. RAS deals with transactions varying in duration and complexity. Without good performance, RAS could lead to unsatisfactory services or even corporate loss. As indicated in Table I, the performance can be refined along the time and space dimensions.

Response Time. Speeding information processing will shorten the duration of transactions, thereby improving the performance of RAS.

Memory Usage. The performance can be affected if the RAS uses the internal computing memory extensively. Making efficient use of external computer storage, on the other hand, enhances system performance.

Integrity. RAS must ensure accuracy and consistency of the data to maintain the business standards. Note that business standards are the necessary authorities and routines in the business practice. They are used to ensure that the business organizations regulate the business processes.

Persistence. RAS is required to store and retrieve the state as data in non-volatile storage. Persistence refers to the characteristic of state that outlives the process that created it. Such a characteristic must be accounted for as an architectural decision.

While other NFRs, such as security factor, may be necessary to be considered for an RAS, the above list of NFRs is sufficient for an illustrative purpose. However, the stakeholders need to elicit the key NFR drivers for their particular EIS instead of adopting existing NFR standards without any modification [27]. To identify a driving NFR, one might justify whether or not the change of this NFR will have an impact on the whole software architecture [3].

### B. Eliciting Scenarios

In this section, a scenario refers to the quality attribute scenario. A scenario is an abstracted description of system to be designed; both the user and designer's perspectives have to be considered. Scenarios play an important role in requirements elicitation and analysis.

Scenarios are frequently used in the system development process, e.g., use case scenarios are a part of the rational unified process and the primary sources of the definition of requirements in the agile software development.

When dealing with subjective concepts like NFRs, scenarios can be used to evaluate if a set of subjective attributes can be satisfied by software architecture. For this reason, the quality attributes scenarios are created to evaluate the interactions of system from the perspectives of the subjective attributes [15]. In contrast to the terminology scenarios used by others, the scenario here must relate to subjective attributes.

In other words, NFRs have to be defined clearly in a scenario. For example the statement “a system is flexible' is invalid since it is vague and meaningless. All systems are flexible to accommodate a certain type of changes [28]. On the other hand, the following statement will be valid for a scenario:

“A user expects to insert an editable field for searching and add an active in the graphical user interface; the icons in the toolbar must be scaled, and the changes should be completed within 3 hours; these changes address the issues 4 and 12 raised in the bug report so that usability will be improved.”

The scenarios make NFRs measurable and also help resolve terminological ambiguities by capturing the stakeholders' precise concerns. For an RAS, we devise the following scenarios.

Sce1: To meet the system requirements, RAS developers take into account the organizational workload. Some transactions (e.g., Submit\_Expense\_Report) are invoked more frequently than others (e.g., Foreign\_Exchange). Caching frequently occurring and business-critical transactions will improve the response time and persistence, though the memory usage is likely to experience extra overhead.

Sce2: A temporal constraint of RAS is that a participator is not allowed to attend a meeting unless he or she registers it. RAS administrators want efficient enforcement of integrity constraints so that unsatisfied business rules can be detected, monitored, and eventually corrected. However, searching and checking every constraint at every point in time can negatively affect system performance.

Sce3: When planning a research-related business trip, the users would like RAS to interact with a variety of external services like flight, hotel, and car rental so that the main memory usage can be reduced. Not only shall RAS ensure local integrity constraints, but the complex cross-organizational constraints such as “the travel expense must be less than the maximized amount specified for each research project” must also be enforced.

There are other factors be considered. For example, a scenario should associate the tasks with the roles of participators [15]. In this way, system usages can be evaluated from multiple perspectives: software developers in Sce1, system administrators in Sce2, and end users in Sce3. The other factor is the use of templates; although a few of templates have been introduced [3], the proposed scenario used general descriptions instead of formatted structure. In Table II, the relationship of the scenarios and the NFRs is specified on a qualitative rating scale. The check mark and cross mark have shown the positive and negative contributions of the scenario to the NFR, respectively, and the field with ‘N/A’ has shown that the NFR is not explicitly considered in the scenario [18].

### C. Evaluating of System Architecture

Tradeoff on the conflict objectives must be made in an engineering situation that involves competing contingencies. Tradeoff on software architecture is about how to make the decisions with a full comprehension of both the upside and downside of a particular choice. We examine two software architecture alternatives for RAS.

The first is database-centered architecture presented in Fig. 2, where the design considerations are arranged in a grid. The horizontal axis depicts the types of conceptual or ontological features involved in an RAS: Entity, Activity, and Constraint. The vertical axis describes the features of the organizations in a semantic model: Classification, Aggregation, and Specialization. The other narrative contents in the plot have shown how the objectives of RAS are translated and related to the system components. The architecture, as shown in Fig. 2, is a layered structure where the information flow occurs only in two adjacent layers. This modular property of information hiding, together with the centralized data model, makes the database-centered architecture a suitable support for the caching organizational workload requirement in Sce1 and the checking local integrity constraints requirement in Sce2.

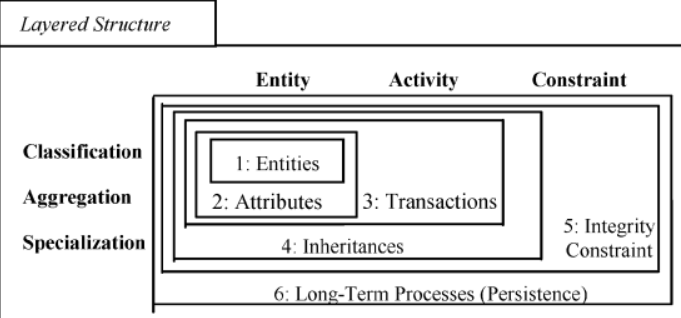


Figure 2: Database-centered layered architecture

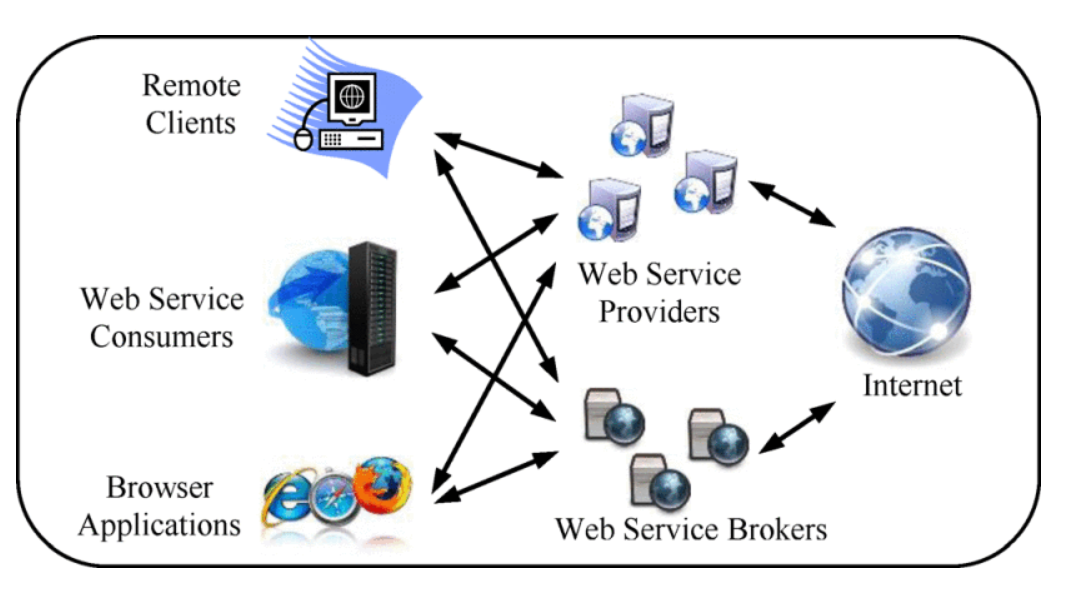


Figure 3: Web services architecture

The second option of software architecture for RAS, as illustrated in Fig. 3, is a distributed service-oriented infrastructure that exploits the Web services. In a nutshell, the interactions among the machines are supported by a web service, which is WSDL-based machine-processable interface. As a result, the Web services architecture is particularly suited for supporting the remote communication use case described in Sce3. At the same time, the caching need stated in Sce1 can be supported by client-side applications.

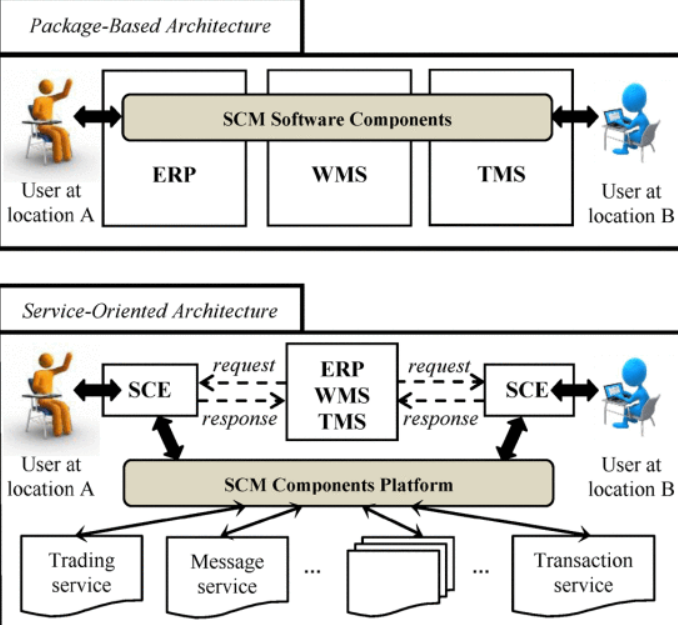


Figure 4: BXD's SCM software architecture alternatives

Having teased out what scenarios are explicitly supported by each design alternative, software architecture options can be evaluated. The main idea is to use the scenarios to connect software architecture with the driving NFRs and to propagate the contribution relations through qualitative design reasoning. For reasoning, the evaluation is focused on finding a solution that is sufficiently good rather than fully optimal in all aspects. The underlying rationale of reasoning in such a matter is that the satisfaction of NFRs is not a straight true or false answer [18]. Take Table II as an example, no matter which scenario or combination of scenarios are supported, there is always one or more NFRs that are negatively affected. In another word, the optimal architecture that meets all of the RAS stakeholders' needs is simply non-existent.

Effective design analysis under the circumstances like the design of an RAS, in our opinion, depends on the ability to aggregate and present the relevant information in an insightful way to help decision makers to find the right balance among the NFRs. We use an intuitive and integrated graphical representation to serve this purpose.

**Order management process in coca cola**

Brief explanation & challenges

* In 2014, Coca-Cola had refranchised and due to which the sales increased a lot and expanded into different new areas.
* They introduced Coca-Cola freestyle which was a self-serve dispenser which was deployed to fast food stores and other suppliers, it offers more than 100 products of coca cola for example sprite, bottled water etc. But this complicated the distribution system as “Coca-Cola innovation collided with their supply chain system”.
* Below figure shows the order management process used by Coca-Cola:

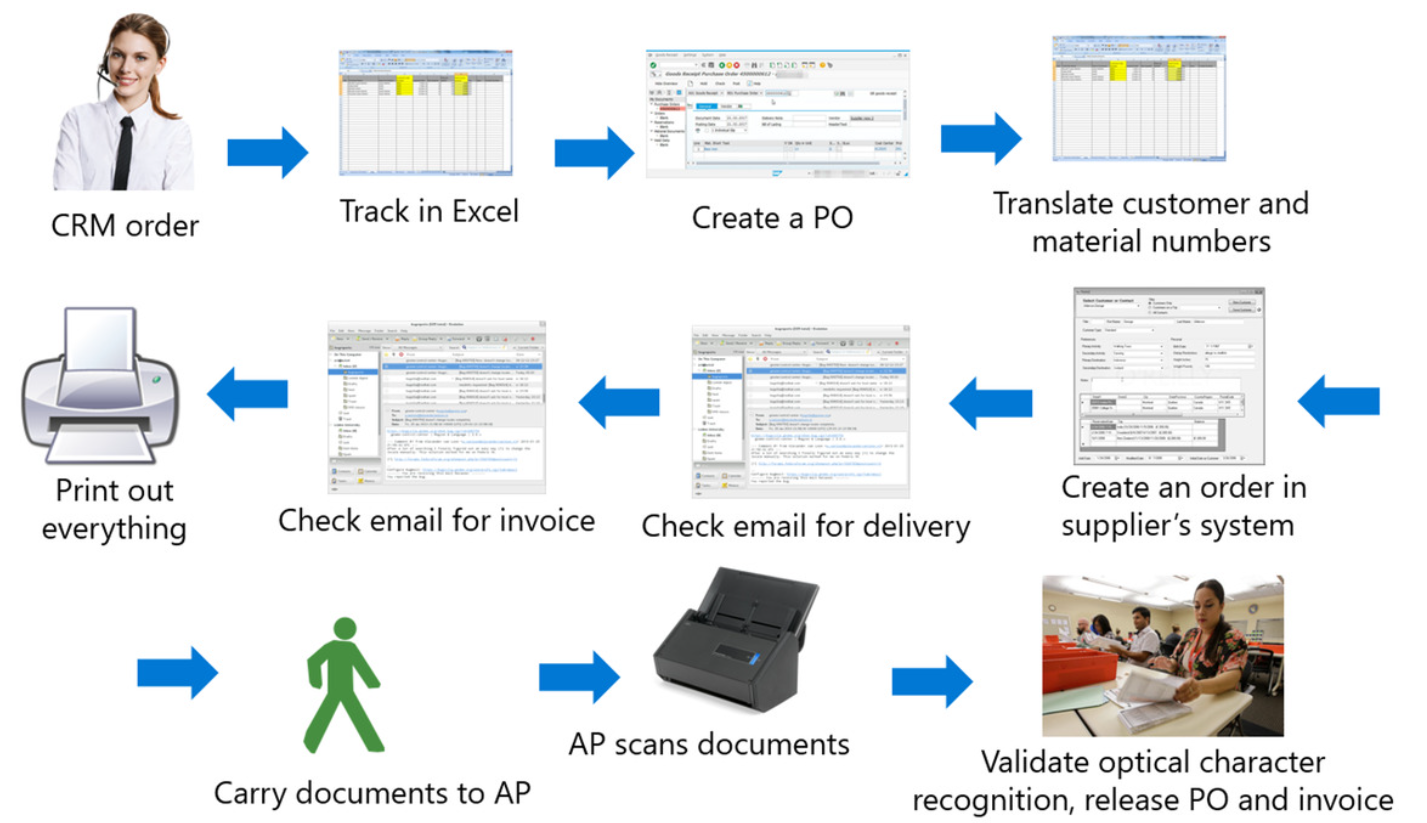


Figure 5: Manual order management process before implementing RPA

* The order management process consisted of 11 steps, at first a CRM order is received which Coca-Cola stored in excel sheets like the Customer and material numbers. PO was created manually after agent take order and push manually through CRM system, then scanned the invoice and walked to accounts payable. Internal numbers were kept in SAP system by Coca-Cola. Delivery and invoice emails were tracked manually by email.
* A single CRM agent can handle only 50 Freestyle orders, and nearly 50,000 more used to be in waiting. These problems were solved by Coca-Cola by implementing RPA with Microsoft Power Automate”.

**New order management process**

Technology & Tools

Microsoft Power Automate helps organizations build multi-stage workflows which automate repetitive business processes to help employees save time and increase productivity. With template workflows available for many common processes, the capability to create custom workflows from scratch, and integration with hundreds of systems and applications, Power Automate offers the flexibility to meet a diverse range of needs. Like the rest of Power Platform, all of this can be accomplished via intuitive, user-friendly interfaces which do not require technical knowledge to use, allowing your organization to save time, increase productivity, and reduce the potential for human error. Though basic Power Automate functionality is accessible to individuals without development expertise, engaging a trusted partner to help you develop your workflows enables organizations to leverage advanced functionality involving complex, multi-system workflows and fully custom flow parts. Data prise DevOps offers the knowledge and experience you need to unlock the full potential Power Automate has to offer and securely automate your business processes at scale. (Power, B.I., 2020)

Architecture

Business Improvements

Advantages and Disadvantages

The company faced two challenges as it explored solutions. “It was too costly to create APIs for our legacy applications like SAP and others, and we also had to navigate a third-party website,” explains Allan McDaniel, Manager of Development for BI and Master Data at Coca-Cola Bottling Company United. “We used Power Automate RPA bots to fill these gaps and automate the process.” Desktop flows in Power Automate automate repetitive processes in Windows and web applications—a perfect fit for high-volume but mundane data entry and transfer.

The solution was built by the company’s fusion teams of citizen developers and professional developers and its partner, Happiest Minds Technologies. Coca-Cola United started by synchronizing data between the company’s SAP CRM system and Azure SQL using Azure Data Factory. Happiest Minds added the automation, creating a master automated service agent they’ve dubbed “Asa.” Developed on Microsoft Azure and Microsoft Power Platform, Asa consists of several bots and uses Azure Key Vault to help secure and control passwords and other sensitive data.

Power Platform Build Tools for Azure DevOps were used to co-ordinate work between Coca-Cola United and Happiest Minds, making it easier to drive continuous improvement and development across the project. These tools also enabled automation of common build and deployment tasks. “Power Platform and Azure DevOps enabled our citizen developers, pro developers and partner to build as a team – and that accelerated the entire development process,” says Means.

Now, when a CRM agent enters an order into the CRM system, Asa takes it from there and signs in to the company’s SAP system without human intervention. Asa easily accesses orders, which are now tracked in a [Microsoft Azure SQL database](https://protect-us.mimecast.com/s/gTGOC9r70Ys3rZxCEANfU) rather than in an Excel spreadsheet. Asa reads the database and creates a PO in the company’s SAP system. Asa then submits the order to the supplier’s web application, validates successful entry, monitors the email system for invoice and delivery emails, matches them to the correct order, and then stores the attachments in [Azure Blob Storage](https://azure.microsoft.com/en-us/services/storage/blobs/) for future reference. After that, Asa uses form processing in [AI Builder](https://flow.microsoft.com/en-us/ai-builder/) to extract information from those email attachments that’s necessary to close the process in the Accounts Payable system, and finally, it releases the invoice and PO from SAP. These steps occur with bots within the Asa bot running in unattended mode, a key capability of Power Automate RPA that takes mundane, tedious tasks from humans and shifts them to bots. With RPA unattended mode, everything is fully automated. After deploying the unattended mode in [Azure Virtual Machines](https://azure.microsoft.com/en-us/services/virtual-machines/), Coca-Cola United can now schedule and trigger events that increase end-to-end automation of high-volume tasks—like its suddenly expanded orders for Freestyle. And because Asa is cloud-based, it can automatically scale to any job and interoperate with any application. Just as importantly, Power Platform and Azure provide a rich set of monitoring and alerting capabilities that facilitate debugging.

The new, simplified process frees the dedicated CRM agent, allowing orders from all channels, such as inbound and outbound call center agents, field service sales representatives at customer sites, and via a customer self-service portal.

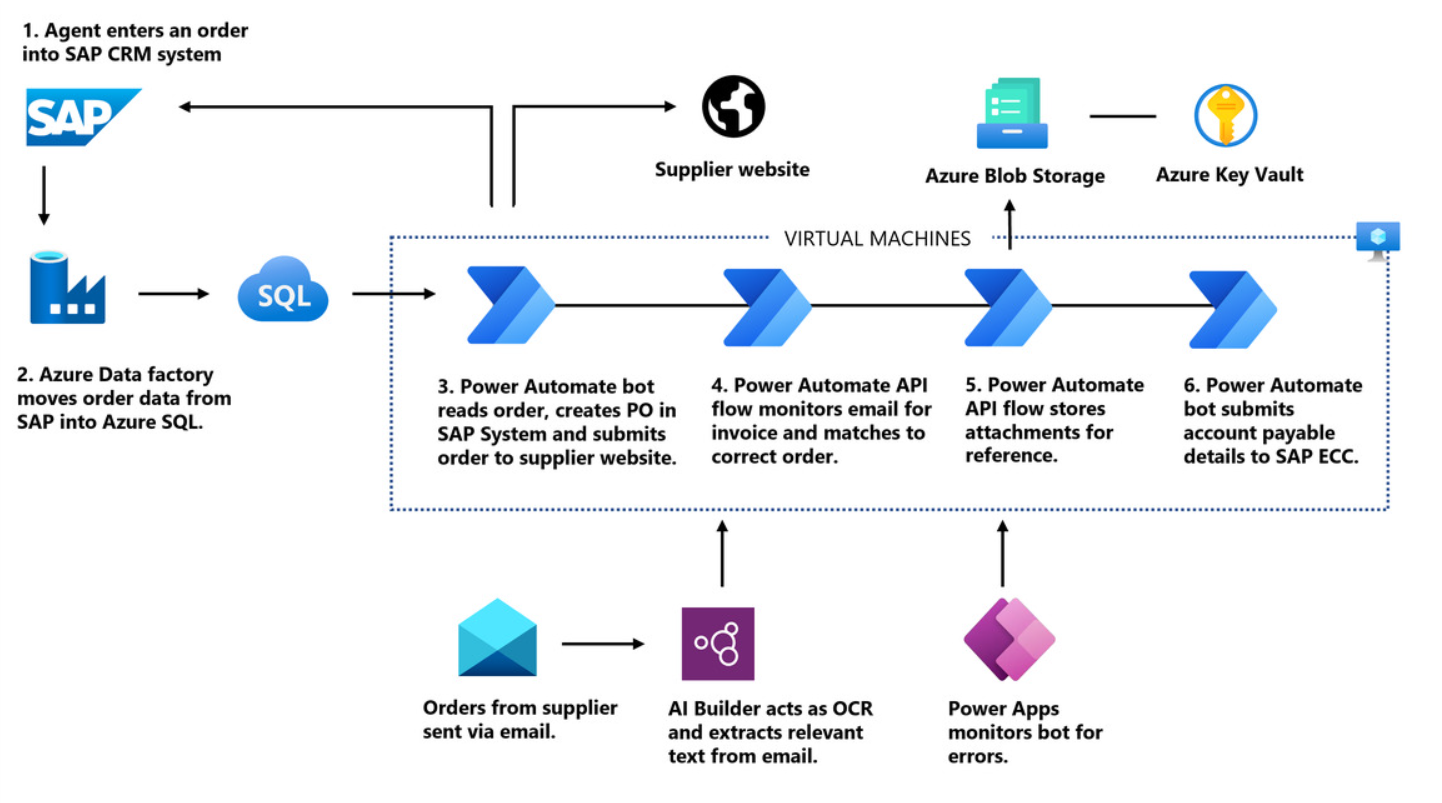


Figure 6: New automated process of order management system using Microsoft Power Platform

By removing the busywork of order and invoice processing, Coca-Cola United vaulted over the limitations it faced prior to creating its Asa bot. ”We avoided having to hire 10 full-time employees,” says McDaniel. “Better still, the existing CRM agent is now free to work on other projects.” Kaylan Cannon, the company’s Customer Service Manager, is enthusiastic. “We are very excited about this solution,” she says. “It will dramatically reduce labor costs, minimize the various points of error in our current solution, and will allow us to rapidly expand the local Freestyle campaign to better support our customers.”

For McDaniel’s team, Azure DevOps interoperability, with rich enterprise-ready security controls, is the gift that keeps on giving. “The low-code to no-code Power Automate RPA platform with built-in connectors significantly reduces development time,” says McDaniel. “The way that Azure DevOps interoperates with the solution plus the built-in administrative and governance controls in Power Platform is an important feature for us,” he adds. “We connect it not only with the Power Automate solution but also any other infrastructure needed. Once built, we can use a desktop flow in any other Power Automate flows, extending its value to solve other business challenges with legacy applications with little to no further development.” And Azure DevOps boosts collaboration with other bottling companies. “We can easily share this work with other organizations because Power Automate works with Azure DevOps.”

Inspired by the productivity boost they achieved with Power Automate RPA, Means and his team are exploring other parts of the business platform, including [Microsoft Dynamics 365](https://dynamics.microsoft.com/en-us/). Azure DevOps and reusability and governance controls will make it easier to share solutions with other bottlers, but for now, Coca-Cola United is buoyed by the joint effort between itself, Microsoft, and Happiest Minds that resulted in an optimal solution, and the improvements continue. “While building this solution, we resurrected high-value strategic projects that we couldn’t tackle before because of the constraints of legacy apps,” says Means. “We feel empowered to take advantage of any future opportunities that the business provides us.”

**Recommendations**

1. There are lot of other RPA tools in the market, which Coca-Cola can check and use

**References**

Niu, N., Da Xu, L. and Bi, Z., 2013. Enterprise information systems architecture—Analysis and evaluation. *IEEE Transactions on Industrial Informatics*, *9*(4), pp.2147-2154.

Power, B.I., 2020. Microsoft Power Platform.