International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 2
Article Received: *05 January 2022* ***Revised:*** *30 January 2022* ***Accepted:*** *20 February 2022*

Designing Scalable B2B Integration Solutions Using Middleware and Cloud APIs

Rajalingam Malaiyalan

Independent Researcher, USA.

Abstract

The main objective of a number of middleware frameworks and technologies is to facilitate the integration process of distributed systems by promoting interoperability between them. A change in value generation was started with the advent of Industry 4.0. Data-driven services—also referred to as smart services—that use data analysis techniques like machine learning are becoming more and more important for new business models and the inter-company data exchange processes required for them. The present paper analyses the main developing technologies that offer various degrees of interoperability that enable quick application development for e-businesses in the context of business-to-business (B2B) e-commerce. In order to demonstrate the effectiveness of current middleware in resolving integration issues across systems within a particular domain, these technologies are shown via a case study in the global financial sector.

Keywords: - Middleware Technologies, Business-to-Business (B2B), E-Commerce, Integration Process, CPQ Systems, Industry 4.0, Data-Driven Services, Implemented.

I. INTRODUCTION

For a long time, the key to understanding and optimising a supply chain that is becoming more and more internationally dispersed has been the digitisation of business activities and the integration of those processes across business partners [1]. In business-to-business (B2B) operations that include the exchange of tangible goods, well-established strategies like vendor-managed inventory or Just-in-Sequence/Just-in-Time processes boost productivity and efficiency rates [1, 2].

The purpose of the standard software development process is to create a whole application in one go. Applications are packaged as a single deployment artefact that contains all of its modules or components. While this approach is a simple and practical way to get started, it may become a barrier to scaling [3, 4]. This development process renders the program inflexible and hard to modify later. Many software programs struggle to adjust to the dynamics of growth and agility in real-world scenarios.

Usually, the architecture of these apps is altered to accommodate the needs of scalability and to provide their end users additional features and services. In addition to losing their competitive edge, applications that don't satisfy these standards run the danger of becoming outdated very fast [3, 4]. Web applications may generally be deployed on-site or in the cloud. Since the cloud provider handles the administration, setup, and upkeep of the server hardware, cloud computing is an effective deployment strategy [4, 5].

Virtualisation, which enables many apps to run concurrently on a same computer system and target various operating systems, is a key component used by cloud providers [5, 6]. These operating systems operate in a layer that is isolated from the hardware itself. Virtualisation facilitates effective resource sharing across several operating systems.

Application processes (i.e., components) and end consumers may communicate and work together over a network thanks to middleware, which is a collection of standard business-unaware services. The software that sits above the network and underneath the business-aware software applications is known as middleware [6, 7]. The best middleware should conceal the variations in computing systems, programming languages, and network protocols in addition to the quality features like real-time, safety, security, and performance.

In other words, the ideal middleware would enable content and process communication across scattered autonomous systems as if they were parts of the same system. Middleware is not yet developed sufficiently to accommodate more complex interoperability abstractions between applications and systems [6, 7]. These interoperability abstractions are part of the e-

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 2
Article Received: *05 January 2022* ***Revised:*** *30 January 2022* ***Accepted:*** *20 February 2022*

commerce framework and are designed to facilitate integration between and across various B2B infrastructures.

Three B2B architecture types were mentioned: federated, dynamic (on-the-fly), and centralised. In the first form, the worldwide business process is controlled by a central body, such as an organisation. This kind emphasises process efficiency by establishing a close-knit, long-term, and static connection between participants (i.e., partners). The second kind lacks a centralised control body but nonetheless emphasises process efficiency [8, 9]. Static, long-term, and loosely or closely connected relationships are supported. The last kind emphasises transaction efficiency by facilitating short-term, loosely connected, and transitory interactions among participants.

1.1 B2B Integration

Giving the interacting entities—such as customers and sellers—a consistent picture of the e-commerce system is known as business-to-business (B2B) integration. Connecting front-end and back-end systems is another aspect of business-to-business integration. These systems may consist of external partner systems or older data sources and programs [4, 6]. In the age of Web-based e-commerce, B2B integration is a difficult undertaking due to the following reasons:

- There is a growing variety of information forms.
- The information space is numerous and ever-changing.
- Data semantic integration is still a work in progress [1, 9].
- The majority of systems are self-sufficient, meaning that their architecture is not centralised.

Quick, easy, safe, and change-adaptable integration is required. Integration may be carried out utilising a variety of middleware technologies at different e-commerce system levels [8, 9]. Integration is offered at the following several levels by the B2B Integration Framework that is covered in Figure 1:

The message exchange between parties is the focus of the Communication Layer (Transport). A variety of protocols and frameworks, ranging from distributed object frameworks to network-level protocols for communicating, have been established [4]. By translating and converting messages across diverse protocols, this layer's interoperable goal is to provide independence from such protocols and environments. Semantic and structural heterogeneity problems are resolved by the Content Layer (Data).

For instance, it ascertains if a document is a purchase order, quotation request, product description, etc. Different information formats are used, which results in structural variations. Variations in how the same notion is interpreted give rise to semantic disparities. For instance, a data item named "price" may refer to a price that contains or does not include tax. Thus, the interoperability goal of this layer is to provide independence from formats, languages, and data structures [9]. Information translation and integration constitute the foundation of solutions; mediators and wrappers are two examples.

The Business Process Layer, also known as Process Flows, handles the semantics of interactions that relate to collaborative business processes. A collaborative business process may include, for instance, sending an order, processing it, delivering the goods, and paying for it. Issues like what a message means, what may be done, what is expected of you, etc., are resolved by this layer. Thus, the goal of the layer's interoperability is to enable transparent peer-to-peer communication with any partners. This is a really challenging task. Potential solutions include document-based solutions, workflow-based solutions, and the Application Programming Interface (API) [11].

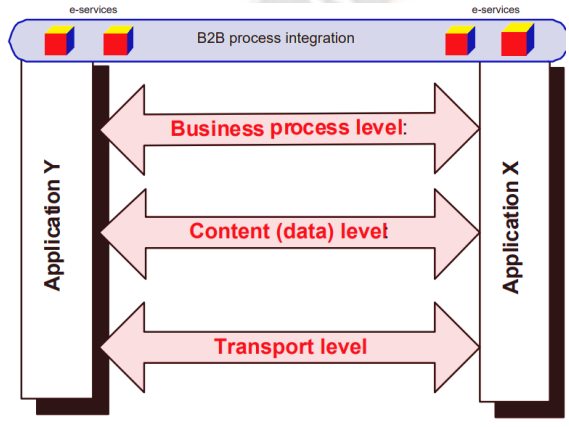


Fig. 1 Layers of B2B interoperability. [10, 11]

Configure, Price, Quote (CPQ) software is a sales enablement solution that helps companies easily create expert quotations, establish precise pricing, and configure complicated items. By automating the labour-intensive processes of product setup, pricing, and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 2
Article Received: *05 January 2022* ***Revised:*** *30 January 2022* ***Accepted:*** *20 February 2022*

quotation preparation, CPQ improves accuracy and streamlines sales operations. Because it makes it easier to create personalised quotations and contracts, this software is especially helpful for businesses that provide highly adjustable goods or services [22]. In the fiercely competitive and fast-paced SaaS sector, businesses must successfully grow their operations and innovate constantly, among other problems.

For SaaS organisations that often deal with intricate pricing models and product customisations, CPQ software solves these issues by automating and streamlining the sales process [21, 22]. Because of its versatility in accommodating several pricing models, including value-based, subscription-based, and usage-based pricing, CPQ is a vital tool for SaaS companies. By facilitating quicker and more accurate quotation production, CPQ software not only shortens the sales cycle but also enhances sales effectiveness with improved data analytics and assisted selling capabilities.

It additionally assists in minimising mistakes in product setups and bids, which lowers revenue leakage and increases customer satisfaction [9, 10]. The integration complexity of CPQ software implementation in a SaaS environment is substantial. Several departments, including Sales, [10,12], Finance, IT, and Product Management, must work together to ensure a successful rollout. A uniform source of truth for customer data, product catalogues, and price information must be maintained during the integration process, which must guarantee smooth interaction with current systems like CRM and ERP applications [4].

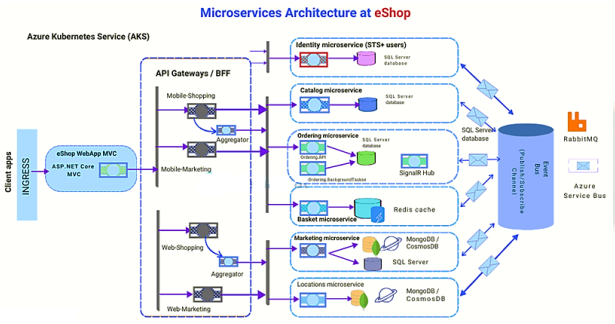


Fig. 2 CPQ services architecture based on microservices. [11]

II. MICROSERVICES ARCHITECTURE

In the context of CPQ, microservices architecture refers to a design methodology in which the CPQ system is divided into smaller, independently deployable services, each of which is in charge of a distinct business capability [13, 14]. The following fundamental ideas form the basis of this architectural style:

- **Service independence:** It is possible to individually build, implement, and grow each microservice.
- **Loose coupling:** Well-defined APIs allow services to communicate with one another, reducing inter-service dependency [14].
- **Single responsibility:** Within the CPQ process, each service focusses on a certain business function.

There are three layers in the client-server paradigm of Docker: the registry, Docker client, and Docker host. The architecture of Docker is explained in detail in Figure 3. Otherwise, the end user cannot communicate with the docker daemon; only the docker client is permitted to do so. The end user, who may not be on the same computer as the daemon, uses the docker client to deliver a command. Through Rest API interfaces, the docker daemon receives the command. After that, the Docker daemon runs the command and returns the result [14]. The daemon is in charge of constructing, managing, and allocating the containers.

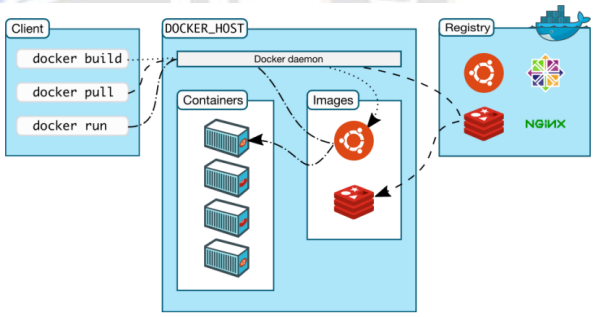


Fig. 3 Docker architecture. [15]

Software systems are traditionally built using monolithic architecture. The fundamental design idea behind many software systems that are now in use across several sectors and enterprises is this architecture. All software components are intended to be assembled into a single, sizable piece using monolithic architecture. Due to its interdependencies and interconnections, monolithic software often lacks modularity among its many components. The bounds of modularity for these components are not clear. In the end, [15], these disparate parts or modules are deployed together as a single process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 2
Article Received: *05 January 2022* ***Revised:*** *30 January 2022* ***Accepted:*** *20 February 2022*

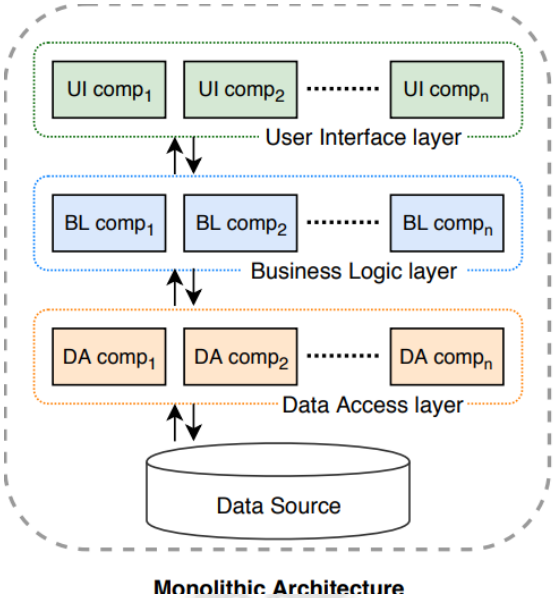


Fig. 4 Typical monolithic application layered architecture. [14, 16]

III. TRANSPORT LEVEL MIDDLEWARE

We want to refer to an integrating process based on the lowest levels of interconnection as low-level middleware [11]. The network protocols that are used to transmit raw data and the remote procedure call (RPC), which is the first form of the distributed computing architecture, are the two ideas that we will cover at this stage.

3.1 Network-protocol level

Early distributed systems were constructed with limited choices in order to accomplish interconnectivity [11,12]. These choices use programming at a relatively low level and are more closely tied to the underlying network protocols.

3.2 Object and component-based middleware Object

A majority common strategies for creating distributed systems are component-based and orientated methods. Because it facilitates closely linked connection, the various middleware types discussed in this section are most appropriate for the centralised design of B2B distributed systems. This middleware is not a suitable option for inter-enterprise connection because of its closely linked connectivity, which requires that the connecting components understand each other's object/component interface [14, 15].

3.3 Content level middleware

A data transfer format specifies how the data should be formatted so that multiple parts can exchange messages, data, and even documents that they can comprehend and interpret—not just within the same distributed business system, but also across systems that are located in different companies and technologies [15]. Initially, the common format should be agreed upon by both the sender and the recipient. Typically, the sender component contains software (an embedded part) that encodes the message's contents into a certain format before sending it to the distant object or component via middleware protocols [17, 25].

3.4 Electronic Data Interchange(EDI) based integration frameworks

In the industry, Electronic Data Interchange (EDI) is well-established and well-defined as it offers both specified vocabularies and syntax for the contents of the EDI message fields [17]. Using EDI standards, suppliers exchange business documents over value-added networks (VANs). Long before the Internet was created, EDI was in operation.

3.5 XML-based frameworks

An emerging collection of open standards for information creation and consumption, the Extensible Markup Language (XML) is overseen by the World Wide Web Collective (W3C) [18]. A lightweight subset of the Standard Generalising Markup Language (SGML) serves as the foundation for XML, a data-oriented technology that may be used for the creation, [19], storing, and retrieval of structured data. Language independence is intrinsic in XML.

IV. BUSINESS PROCESS MIDDLEWARE

Descriptions of the actions necessary for an organisation to accomplish its goals, including fulfilling a business contract or a particular consumer request, are known as business processes [19, 20]. Taking charge of these procedures enables a company to modify them to meet evolving needs or reengineer and enhance each one.

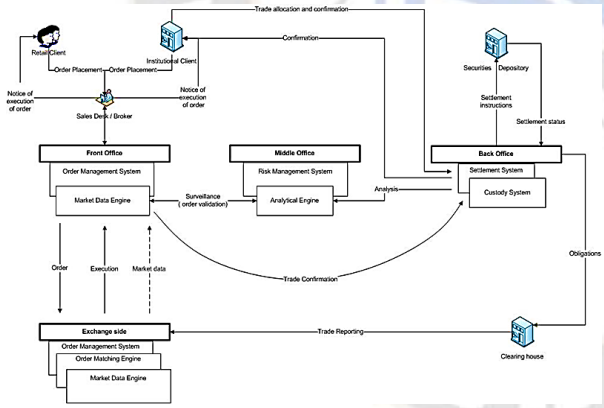
4.1 Application Programming Interface (API)

involves figuring out business processes offline, figuring out how everything is connected and coordinated, and then creating widely accepted abstract interfaces that provide back-end system connections and remote operation invocations [21]. These abstract interfaces are

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 10 Issue: 2
Article Received: *05 January 2022* ***Revised:*** *30 January 2022* ***Accepted:*** *20 February 2022*

subsequently translated into physical implementation using middleware and database technologies. CORBA-based solutions are one example of such methods.

- Document-based remedies a protocol governs the exchange of a collection of documents. Since partners individually publish their publications, there is no previous agreement.
- The foundation of traditional workflow systems is the idea that managing business processes holistically is essential to an organization's success. In fact, a growing number of organisations have already used workflows to automate their internal process management, and they have reaped significant advantages from doing so [22].
- A loosely connected, document-based integration architecture for Internet-based applications, the Web Services concept is gradually gaining traction [17]. Web services provide inter-enterprise workflows, which differ from conventional workflows in that they allow business processes from different organisations to communicate loosely connected via the exchange of XML document-based messages.



Rig. 1 Cycle of Securities Trading. [17]

V. CASE STUDY: CAPITAL MARKET SYSTEMS INTEGRATION

The recent technical advancements that have enabled the immediate interchange of information, assets, and money globally have made trading in capital markets like the New York Stock interchange and the Australian Stock Exchange a more global activity [18]. Decisions about trading have become more complex due to the fact that they are made by several individuals interacting over large geographic distances and time zones.

The conventional communication medium is sluggish and ineffective, which results in a significant loss of potential trade possibilities [19]. The telephone, fax, and email are examples of traditional communication methods for multiple platforms within the same market or across various industries because the majority of these systems are not integrated in a way that enables business processes to move seamlessly between systems without the need for human interaction.

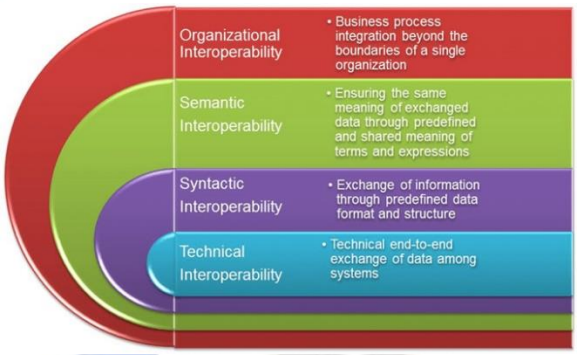


Fig. 5 The degree of interoperability in capital market systems. [22]

The majority of financial systems use private or dedicated networks to communicate with one another for security concerns. The SWIFT Transport Network (STN), which uses the X.25 protocol, is one example. Considering that communication software is not standardised, this is a very costly approach [24, 23].

Many international institutions were already attempting to embrace common standards in the financial sector at the same time as XML advancements. The Financial Information e-Xchange (FIX) protocol is one of these protocols for sharing securities transactions between two parties.

Fig. 6 Essential services related to trading securities. [24]

A basic service or a composite (integrated) service is offered by each software component. In the former case, the architectural element provides the service logic in its

entirety [18]. In order to complete its function, the latter is where the architectural component interacts and needs additional services, often known as outsourced services.

VI. CONCLUSION

The three layers of interoperability of B2B e-commerce applications—transport (low level and object and component level), contents, and business process levels—have been covered in this study. We have seen how several middleware systems facilitate interoperability at various levels throughout the article. However, none of the three tiers are supported by any middleware technology. The interoperability support for the main middleware technologies is briefly discussed here:

- Low-level communication at the communication level, middleware facilitates ad hoc interoperability. At the lowest level, loosely connected interaction is supported via network-level TCP/IP. The first type that provide closely linked interaction is RPC.
- In the context of closely connected services, component and object-based frameworks emphasise compatibility at the communication layer. They are more suited for integrating intra-business services.
- EDI is primarily concerned with interoperability at the content and communication levels when it comes to loosely connected services. This method works well for combining inter-enterprise services with stable, long-term commercial relationships.
- Various XML-based frameworks offer interoperability at various levels. They often concentrate on interoperability at the content and communication levels when discussing loosely connected Internet-based services. RosettaNet and ebXML support for interoperability at the business process layer is limited.
- While new initiatives in this field seek to enable all interoperability levels in the context of inter-enterprise services, workflow-based solutions concentrate on interoperability at the business process layer in the context of closely connected services.

In general, we have spoken about the technologies that facilitate e-commerce. We have concentrated on the limitations these technologies have on software developed using them as well as the interoperability they enable. There will be greater issues with interoperability and integration as the number of these technologies grows without a clear leader. This would compel rival technologies to provide ways to integrate with other important technologies. Because it adds extra adapter layers to ensure interoperable across applications with various middleware architectures, this approach has a performance costs.

VII. REFERENCES

[1] Bhardwaj, A. Deshpande, A. J. Elmore, D. Karger, S. Madden, A. Parameswaran, H. Subramanyam, E. Wu und R. Zhang, „Collaborative Data Analytics with DataHub,“ Proceedings of the VLDB Endowment, Bd. 8, Nr. no. 12, p. 1916–1919, 01 August 2015.

[2] Bundesministerium für Wirtschaft und Energie (BMWi), „Smart Service Welt: Internetbasierte dienste für die Wirtschaft,“ 2017.

[3] Bundesministerium für Wirtschaft und Energie (BMWi), „Erfolgreich Smart Services Entwickeln: Projektabschlussbroschüre der Smart Servie Welt I,“ 2019.

[4] R. Wirth und J. Hipp, „CRISP-DM: Towards a Standard Process Model for Data Mining,“ in Proceedings of the Fourth International Conference on the Practical Applikation of Knowledge Discovery and Data Mining, 2000.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo und D. Dennison, „Hidden Technical Debt in Machine Learning Systems,“ Advances in Neural Information Processing Systems 28, 2015.

[6] Graziano, C. D. A performance analysis of xen and kvm hypervisors for hosting the xen worlds project. PhD thesis, Iowa state university, 2011. https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3243&context=etd.

[7] Harter, T., Salmon, B., Liu, R., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. Slacker: Fast distribution with lazy docker containers. In Proceedings of the 14th Usenix Conference on File and Storage Technologies (Berkeley, CA, USA, 2016), FAST'16, USENIX Association, pp. 181–195.

[8] Jaramillo, D., Nguyen, D. V., and Smart, R. Leveraging microservices architecture by using docker technology. In SoutheastCon 2016 (March 2016), pp. 1–5.

[9] kubernetes.io. Getting Started - Container runtimes. https://kubernetes.io/docs/setup/production-environment/ container-runtimes/. Accessed 29 Oct 2019
[10] Taibi, Davide, Valentina Lenarduzzi, and Claus Pahl.
[11] "Architectural Patterns for Microservices: A Systematic Mapping Study." Closer (2018): 221-232.
[12] Guide, Solutions, Dipanker Jyoti, and James A. Hutcherson. "Salesforce Architect's Handbook."
[13] Baldwin, Donald. "A Domain Neutral Enterprise Architecture Framework for Enterprise Application Integration and Pervasive Platform Services." (2015).
[14] Gerald Brose, Andreas Vogal, and Keith Duddy. Java Programming with CORBA: Advanced Techniques for Building Distributed Applications. John Wiley & Sons, Inc, 3rd edition, 2001.
[15] A. W. Brown. Large-Sclae, Component-Based Development. Prentice Hall, 2000.
[16] Ritson, Carl G., and Peter H. Welch. "A process-oriented architecture for complex system modelling." Concurrency and Computation: Practice and Experience 22, no. 8 (2010): 965-980.
[17] Christoph Bussler. B2b protocol standards and their role in semantic b2b integration engines. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24(1), 2001.
[18] Feras Dabous and Fethi Rabhi. Smarts architecture and performance. Technical report, School of ISTM, The University of NSW, June 2002. confidential document that concluded seed grant from CMCRC.
[19] Research information exchange markup language web site.
[20] The simple object access protocol (soap) specification web site.
[21] M. Eisenträger, C. Frey, A. Herzog, A. Moghiseh, L. Morand, J. Pfrommer, H. Stephani, A. Stoll und L. Wessels, „ML4P-Vorgehensmodell: Machine learning for production".
[22] Morabito, R., KjAllman, J., and Komu, M. ˜ Hypervisors vs. lightweight virtualization: A performance comparison. In 2015 IEEE International Conference on Cloud Engineering (March 2015), pp. 386– 393.
[23] Hasterok, J. Stompe, J. Pfrommer, T. Usländer, J. Ziehn, S. Reiter, M. Weber und T. Riede, „PAISE - Das Vorgehensmodell für KI-Engineering," 2021.
[24] M. E. Porter und J. E. Heppelmann, „How Smart, Connected Products Are Transforming Competition," Harvard Business Review, November 2014.
[25] S. Leminen, M. Rajahonka, M. Westerlund und R. Wendelin, „The future of the Internet of Things: toward heterarchical ecosystems and service business models," Journal of Business & Industrial Marketing, Bd. Vol. 33, Nr. No. 6, pp. 749-767, 2018.