**Literature Review: Using the Microservice Architecture to Support Web Applications in the E-Commerce Industry**

**Introduction**

The rise of computers has facilitated the creation of new industries. E-commerce is one of the largest of these, with its market having been valued at $10.3 trillion in 2020 (Grand View Research, 2020). Software engineering drives this industry, thus good software architecture is critical. Industry and academia have developed various perspectives on the topic of software architecture within e-commerce, having focused on two specific types- monolithic architectures, and microservices. Discrepancies may exist between industry and academia; the aim of this article is to therefore conduct a literature review and develop an understanding of what the current state of the art is, for using microservices to develop e-commerce web applications.

The methodology for this literature review uses engineering focused resources. The overall approach is therefore adapted from the methodology discussed by Torres-Carrión et al. (2018).

To begin the review, research questions are formulated by applying the PICOC and CIMO methodologies (Ermel et al., 2021).

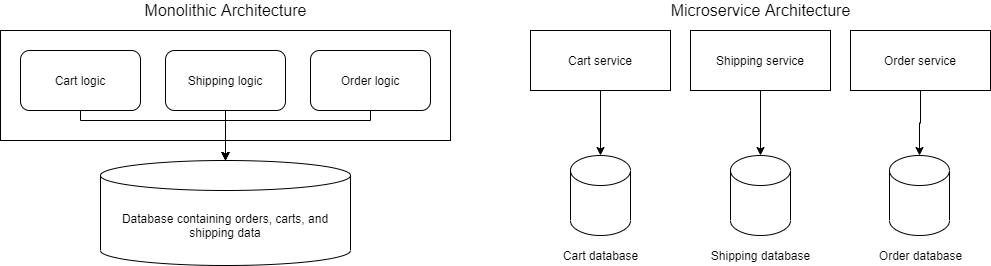
Thereafter, databases are searched for articles directly related to microservices in the e-commerce sector. This literature review uses articles found by searching Google Scholar, the ACM Digital Library, and IEEE Xplore. As this literature review is focused on evaluating current knowledge on the topic, only articles published between 2017 and 2022 are considered.

A thematic analysis is then conducted. To do this, articles are grouped by common themes. Articles on each theme are critically evaluated in relation to each other to establish an overall understanding.

Lastly, findings from the thematic analysis are used to answer the research questions. To conclude the review, research gaps discovered are presented as questions to guide future research.

Research Questions

To determine the research questions by using the PICOC and CIMO methodologies, it is necessary to first frame the questions relative to the predecessor of the microservice architecture, which is the monolithic architecture. The architectures are defined as follows: a monolithic architecture is a software architecture in which an application has all functionality contained in one deployable program, and all required data is stored in one database. In the microservice architecture, functionality is split across multiple deployable programs, with each program having its own dedicated database that isn’t accessed by other applications (Seshagiri et al., 2022). Figure 1 illustrates the differences between the architectures:



**Figure 1:** *A diagram representing a monolithic application compared to a microservice. Rectangles represent deployable programs, while rounded rectangles represent program classes that can’t be deployed independently.*

Based on the above, the following research questions are posed:

**Research Question 1:** How do software engineering practices change in e-commerce when using the microservice architecture instead of the monolithic architecture?

**Research Question 2:** What are the business outcomes for developing e-commerce web applications using the microservice architecture?

**Research Question 3:** When developing a web application for e-commerce using microservices, what functionalities are not handled by microservices when compared to a monolithic application?

**Findings**

Table 1 describes the works found by searching the databases mentioned previously:

| Authors | Agility | Resilience | Performance | Domain Driven Design (DDD) |
| --- | --- | --- | --- | --- |
| Bogner et al. (2019) | ✓ | ✓ | ✓ | ✓ |
| Daraghmi et al. (2022) | ✕ | ✕ | ✓ | ✕ |
| Gan et al. (2019) | ✕ | ✕ | ✓ | ✕ |
| Gördesli & Varol (2022) | ✓ | ✓ | ✓ | ✕ |
| Hasselbring & Steinacker (2017) | ✓ | ✓ | ✓ | ✓ |
| Knoche & Hasselbring (2019) | ✓ | ✓ | ✓ | ✕ |
| Rudrabhatla (2018) | ✕ | ✕ | ✓ | ✕ |
| Suthendra & Pakereng (2020) | ✓ | ✓ | ✓ | ✓ |
| Wu et al. (2019) | ✕ | ✓ | ✓ | ✕ |

**Table 1:** *Describing articles by highlighting themes discussed per author. A tick means the theme was discussed, a cross means that it was not discussed.*

**Thematic Analysis**

The perspectives of various academic authors are now critically evaluated and compared with industry insights to establish a current perspective on each theme.

Agility

Agility can be thought of as the efficiency and speed with which a developer can update, change, and deploy software- this attribute is typically associated with positive business outcomes such as a shorter time to market (Knoche & Hasselbring, 2019).

Microservices have faster release cycles, which increases agility. Release cycles increase agility because changes are isolated to single services, and these services share no dependencies, thus development on different features can be done in parallel (Bogner et al., 2019; Gördesli & Varol, 2022). In comparison, release cycles for monolithic applications are much slower- simple changes can take more time since dependencies are shared, which requires changes to be made in multiple places (Suthendra & Pakereng, 2020).

However, additional work is required to realise this improvement in agility. Since release cycles increase in frequency with the microservice architecture, automation becomes a necessity to support a faster time to market (Knoche & Hasselbring, 2019). Automation is achieved by implementing continuous integration and continuous delivery, which are the practices of automating the testing and release of version-controlled software. In practice, e-commerce developers use both techniques (Hasselbring & Steinacker, 2017; Knoche & Hasselbring, 2019; Bogner et al., 2019).

Resilience

Fault-tolerance (also known as resilience), refers to the ability of a system to continue functioning despite any damage (Suthendra & Pakereng, 2020). Gördesli & Varol (2022) argue that microservices are naturally fault-tolerant because of their distributed and independent nature: for example, a shipping microservice could stop working, but cart and order microservices would remain functional.

Despite this benefit, other literature implies that additional work is needed to make microservices resilient: Suthendra & Pakereng (2020) state that resilience is only guaranteed if monitoring is put into place, while Wu et al. (2019) use a specific technology (Hystrix) to make their microservices resilient. Knoche & Hasselbring (2019) also recommend using Hystrix, but also discuss design patterns that can be used instead.

Industry perspectives on resilience are positive: an e-commerce developer discussed how microservices would often stop working, but users would not notice this because other services remained functional (Bogner et al., 2019); Hasselbring & Steinacker (2017) affirm this finding. An industry whitepaper from SQLI (2020) recommends using microservices to eliminate single points of failure, which matches the academic definition of resilience. Notably, a monolithic application is a single point of failure, since all functionalities are run in one application- if that application stops working, all business functionality is halted.

Performance

To support the fast pace of e-commerce, microservices in the sector must be fast and constantly available (Gördesli & Varol, 2022). However, e-commerce processes have unique requirements which affect the highest performance attainable. Knoche & Hasselbring (2019) suggested that performance may degrade in microservices due to the need for additional network calls. However, Gan et al. (2019) empirically discovered that e-commerce microservices are more compute intensive than microservices in other domains, such as social networking. This means that, in e-commerce, databases and networking are not as influential on performance as the programming language selected.

The distributed nature of microservices adds additional complexity which further affects performance. For example, orders, payments, and shipping must be processed in sequence for correctness, however, implementing sequential processing across services reduces performance (Gan et al., 2019). Guaranteeing data consistency in microservices also requires sophisticated logic; for example, if payment fails, an order shouldn’t actually be created. The saga pattern can be used to enforce data consistency but the performance impact is noticeable- it reduces the number of orders that can be processed per minute (Rudrabhatla, 2018; Daraghmi et al., 2022).

Despite the above, e-commerce developers feel that performance is generally faster with microservices than monoliths, with one developer noting that their microservice response times were much faster and those microservices could support higher traffic volumes; another developer suggested that the increased performance may improve user experience (Bogner et al., 2019; Hasselbring & Steinacker, 2017). Notably, developers in the field do not discuss the performance impacts of sequential processing and enforcing data consistency.

Domain Driven Design

Domain-driven design (DDD) is a technique that is used to determine which responsibilities to assign to an application by considering functionalities common between business processes (Suthendra & Pekareng, 2020). For example, an e-commerce business using microservices might need to take payments from customers and pay shipping companies, thus DDD would dictate creating a payments microservice to handle these processes.

Notably, DDD within microservices doesn’t cater to cross-cutting concerns, that is, functionalities common across all domains- user interfaces are a key example. Microservices don’t provide user interfaces, rather, these are typically implemented as websites managed separately from the microservices themselves (Hasselbring & Steinacker, 2017; Bogner et al., 2019).

The majority of articles found do not discuss DDD, but it has a clear influence since various authors designed microservices which follow the methodology. Gördesli & Varol (2022) created microservices to manage stock, payments, and orders. Wu et al. (2019) created identical services, with the addition of logistics and account services. Suthendra & Pekareng (2020) were the only authors to state that DDD was used to design their microservices, and they created the same microservices as the other authors, with the exception of payments.

DDD is used extensively in industry, however, its implementation is not as simple as academic scenarios, where one business domain is managed by one microservice. In real-world scenarios, business domains may be so complex that they need to be decomposed into subdomains, or multiple unique microservices need to be created to manage that domain (Hasselbring & Steinacker, 2017; Bogner et al., 2019). In contrast, Knoche & Hasselbring (2019) suggest that microservices in the field are actually designed according to the single responsibility principle, however, no other literature supports this.

Lastly, SQLI (2020) noted that a benefit of using DDD with microservices is that different languages can be used to fit the domain being worked on- for example, Python could be used in a data-heavy domain.

**Conclusions**

With the thematic analysis of current literature completed, the research questions are now addressed:

**Research Question 1:** How do software engineering practices change in ecommerce when using the microservice architecture instead of the monolithic architecture?

Various stages of the software development life cycle change when using microservices: during design, domain-driven design is applied to determine which responsibilities to assign to different microservices. During programming, developers are now free to use different programming languages for each service, however, developers should always apply the saga pattern to enforce data consistency. Monolithic applications prevent the use of multiple languages, although they do not require domain-driven design and data consistency is trivial to enforce since all data can be accessed from any function.

In terms of software releases, release cycles are slower for monolithic software since development is slower due to additional complexity in the form of shared dependencies. When using microservices, however, release cycles are much faster because changes are isolated and development can be done in parallel. Manual release strategies cannot support this pace, so companies must implement continuous integration and continuous delivery.

**Research Question 2:** What are the business outcomes for developing ecommerce web applications using the microservice architecture?

Microservices benefit businesses: faster release cycles speed up the time to market. Profitability improves as well- since microservices have no single point of failure, many business processes in different domains can continue to run even if one microservice stops working, unlike monolithic applications, where a single outage can halt all functionality.

Developers in the field feel that microservices provide increased speed and resilience, which benefits the business by improving user experience. However, it is not known whether the high computing requirements of e-commerce pose a threat to this or not.

**Research Question 3:** When developing a web application for ecommerce using microservices, what functionalities are not handled by the microservices when compared to a monolith?

Microservices don’t provide user interfaces, rather, they only provide backend functionality for e-commerce business processes, split by domain. When using the microservices architecture, a dedicated team would be required to build the “frontend”- that is, the visual storefront and web pages that a customer interacts with to complete tasks. These webpages would, in turn, call the microservices to actually execute business processes. In comparison, a monolithic application would have the webpages and visuals built in, which may make functionality easier to reason about since all procedures are linked together. Since microservices undergo constant change, the frontend team would be required to keep web pages up-to-date with newer versions of microservice interfaces.

**Further Research**

Research gaps were identified during this literature review, which can guide future discourse. Research questions are presented below:

* Is a microservice sufficiently resilient on its own because of its independent nature, or are additions necessary to make it truly resilient? If the latter is true, are resilience technologies and monitoring adequate, or are there other additions to consider?
* Domain-driven design is commonly used. Are there any other design methodologies which should be considered when designing microservices?
* Apart from the saga pattern, are there any other techniques used in the field to guarantee data consistency?
* Microservices in e-commerce are compute intensive due to business requirements and data consistency requirements. Do developers think this harms performance, and if so, do they feel that it is worth finding optimisations in these areas?

Overall, it can be said that the microservice architecture provides clear benefits to e-commerce businesses, and adopting this architecture in a strategic manner will empower future growth.

**References**

Bogner, J., Fritzsch, J., Wagner, S. & Zimmermann, A. (2019) ‘Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality’, *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. Hamburg, Germany, 25-26 March. New Jersey: IEEE. 1-9.

Daraghmi, E., Zhang, C. & Yuan, S. (2022) Enhancing Saga Pattern for Distributed Transactions within a Microservices Architecture. *Applied Sciences* 12: 1-24. DOI: <https://doi.org/10.3390/app12126242>

Ermel, A., Lacerda, D., Morandi, M. & Gauss, L. (2021) Literature Reviews: Modern Methods for Investigating Scientific and Technological Knowledge. 1st ed. Cham: Springer.

Gan, Y., Zhang, Y., Cheng, D., Shetty, A., Rathi, P., Katarki, N., Bruno, A., Hu, J., Ritchken, B., Jackson, B., Hu, K., Pancholi, M., He, Y., Clancy, B., Colen, C., Wen, F., Leung, C., Wang, S., Zaruvinsky, L., Espinosa, M., Lin, R., Liu, Z., Padilla, J. & Delimitrou, C. (2019) ‘An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems’, *Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. Providence, Rhode Island, 13-17 April. New York: ACM. 3-18.

Gördesli, M. & Varol, A. (2022) ‘Comparing Interservice Communications of Microservices for E-Commerce Industry’, *2022 10th International Symposium on Digital Forensics and Security (ISDFS).* Istanbul, Turkey, 6-7 June. New Jersey: IEEE. 1-4.

Grand View Research. (2020) E-commerce Market Size, Share & Trends Analysis Report By Model Type (B2B, B2C), By Region (North America, Europe, APAC, Latin America, Middle East & Africa), And Segment Forecasts, 2020 - 2027. Available from: <https://www.grandviewresearch.com/industry-analysis/e-commerce-market> [Accessed 27 July 2022].

Hasselbring, W. & Steinacker, G. (2017) ‘Microservice Architectures for Scalability, Agility and Reliability in E-Commerce’, *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. Gothenburg, Sweden, 5-7 April. New Jersey: IEEE. 243-246.

Knoche, H. & Hasselbring, W. (2019) Drivers and Barriers for Microservice Adoption – A Survey among Professionals in Germany. *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 14: 1-35. DOI: <https://doi.org/10.18417/emisa.14.1>

Rudrabhatla, C. (2018) Comparison of Event Choreography and Orchestration Techniques in Microservice Architecture. *International Journal of Advanced Computer Science and Applications (IJACSA)* 9(8): 18-22. DOI: <http://dx.doi.org/10.14569/IJACSA.2018.090804>

Seshagiri, V., Huye, D., Liu, L., Wildani, A., & Sambasivan, R. (2022) Identifying Mismatches Between Microservice Testbeds and Industrial Perceptions of Microservices. *Journal of Systems Research* 2(1): 1-26. DOI: <http://dx.doi.org/10.5070/SR32157839>

SQLI. (n.d.) Microservices: A Paradigm Shift for Fast-Growing e-Commerce Business. Web: SQLI.

Suthendra, J. & Pakereng, M. (2020) Implementation of Microservices Architecture on E-Commerce Web Service. *ComTech: Computer, Mathematics and Engineering Applications* 11(2): 89-95. DOI: <https://doi.org/10.21512/comtech.v11i2.6453>

Torres-Carrión, P., González-González, C., Aciar, S. & Rodríguez-Morales, G. (2018) ‘Methodology for systematic literature review applied to engineering and education’, *2018 IEEE Global Engineering Education Conference (EDUCON)*. 17-20 April, Santa Cruz de Tenerife, Spain. New Jersey: IEEE. 1364-1373.

Wu, M., Ding, X. & Hou, R. (2019) ‘Design and Implementation of B2B E-commerce Platform Based on Microservices Architecture’, *CSSE 2019: 2019 2nd International Conference on Computer Science and Software Engineering.* Xi’an, China, 24-26 May. New York: ACM. 30-34.