# Instructions

Make sure that all the sources you’ve used in your paper are listed on the reference page. Place the reference page on a new page, right after the body text, but before any appendices.

The Scribbr Citation Generator already applied the APA format to your reference list, meaning:

* The page starts with the section label “References” (bold and centered)
* References are ordered [alphabetically](https://www.scribbr.com/apa-style/ordering-references/?utm_source=citation-generator&utm_medium=word-export)
* The text is double-spaced
* A hanging indent of ½ inch is applied
* Page numbering in the top-right corner

Still have questions? Check out Scribbr’s [article on formatting the reference page](https://www.scribbr.com/apa-style/apa-reference-page/?utm_source=citation-generator&utm_medium=word-export).

Tip: don’t forget to check your [in-text citations](https://www.scribbr.com/apa-style/in-text-citation/?utm_source=citation-generator&utm_medium=word-export) for accuracy. Need a little help? The [Scribbr Citation Checker](https://www.scribbr.com/citation/checker/?utm_source=citation-generator&utm_medium=word-export) can automatically analyze your in-text citations for stylistic errors and inconsistencies, presenting you with simple instructions that help fix them.

# References

*A better domain events pattern · Los Techies*. (2014, May 13). https://lostechies.com/jimmybogard/2014/05/13/a-better-domain-events-pattern/

*Aggregate Pattern | DevIQ*. (n.d.). DevIQ. https://deviq.com/domain-driven-design/aggregate-pattern/

Ashbacher, C. (2010). Succeeding With Agile: Software Development Using Scrum, by Mike Cohn. *The Journal of Object Technology*, *9*(4). https://doi.org/10.5381/jot.2010.9.4.r1

Avram, A. (2007). Domain-Driven Design Quickly. *Lulu.com eBooks*. https://dl.acm.org/citation.cfm?id=1557346

Batista, F. (2022, March 4). *Developing the ubiquitous language - DDD - The Domain Driven Design*. DDD. https://thedomaindrivendesign.io/developing-the-ubiquitous-language/

Betts, D., Dominguez, J., Melnik, G., Simonazzi, F., & Subramanian, M. (2012). *Exploring CQRS and Event Sourcing: A journey into high scalability, availability, and maintainability with Windows Azure*.

Bissi, W., Neto, A. T., & Emer, M. C. F. P. (2016). The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information & Software Technology*, *74*, 45–54. https://doi.org/10.1016/j.infsof.2016.02.004

Braun, S., Bieniusa, A., & Elberzhager, F. (2021). Advanced Domain-Driven Design for Consistency in Distributed Data-Intensive Systems. In *European Conference on Computer Systems*. https://doi.org/10.1145/3447865.3457969

Brewer, E. (2012). Pushing the CAP: Strategies for Consistency and Availability. *IEEE Computer*, *45*(2), 23–29. http://dx.doi.org/10.1109/MC.2012.37

Caron, R. (n.d.). *Get the Azure Quick Start Guide for .NET Developers*. Microsoft Azure. https://azure.microsoft.com/en-us/blog/get-the-azure-quick-start-guide-for-net-developers/

Chinh-Nguyen. (2022). Domain Driven Design in practice - Notes. *vdocuments.net*. https://vdocuments.net/domain-driven-design-in-practice-notes.html?page=1

Cohn, M. (2009). *Succeeding with Agile: Software Development Using Scrum*.

Cwalina, K., & Abrams, B. (2005). *Framework Design Guidelines : Conventions, Idioms, and Patterns for Reusable . NET Libraries*.

De La Torre, C. (2017, February 7). *Domain Events vs. Integration Events in Domain-Driven Design and microservices architectures - Cesar de la Torre*. Cesar De La Torre. https://devblogs.microsoft.com/cesardelatorre/domain-events-vs-integration-events-in-domain-driven-design-and-microservices-architectures/

Debski, A., Szczepanik, B., Malawski, M., Spahr, S., & Muthig, D. (2018). A Scalable, Reactive Architecture for Cloud Applications. *IEEE Software*, *35*(2), 62–71. https://doi.org/10.1109/ms.2017.265095722

Erl, T. (2007). SOA Principles of Service Design (The Prentice Hall Service-Oriented Computing Series from Thomas Erl). In *Prentice Hall PTR eBooks*. Prentice Hall PTR. https://dl.acm.org/citation.cfm?id=1296147

Evans. (2003). Domain-Driven Design: Tacking Complexity In the Heart of Software. *Addison-Wesley Longman Publishing Co., Inc. eBooks*. http://dl.acm.org/citation.cfm?id=861502&dl=ACM&coll=DL

Evans, E., & Evans, E. J. (2004). *Domain-driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional.

Fields, J., Harvie, S., Fowler, M., & Beck, K. (2009). *Refactoring: Ruby Edition*. Pearson Education.

Fowler, M. (2010). *Domain-Specific Languages*. Pearson Education.

Fowler, M. (2012). *Fowler: Pattern Enterpr Applica Arch*. Addison-Wesley.

Garverick, J., & McIver, O. D. (2023). *Implementing Event-Driven Microservices Architecture in .NET 7: Develop event-based distributed apps that can scale with ever-changing business demands using C# 11 and .NET 7*. Packt Publishing Ltd.

Indrasiri, K., & Suhothayan, S. (2021). *Design Patterns for Cloud Native Applications*. “O’Reilly Media, Inc.”

Jamesmontemagno. (2022a, April 13). *Designing a DDD-oriented microservice*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice

Jamesmontemagno. (2022b, September 2). *Containerized Docker Application Lifecycle with Microsoft Platform and Tools*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/containerized-lifecycle/

Jamesmontemagno. (2023, March 22). .*NET Microservices. Architecture for Containerized .NET Applications*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/

Khononov, V. (2021). *Learning Domain-Driven Design*. “O’Reilly Media, Inc.”

Kumar, V., & Agnihotri, K. (2021). *Serverless Computing Using Azure Functions: Build, Deploy, Automate, and Secure Serverless Application Development with Azure Functions (English Edition)*. BPB Publications.

Landre, E., Wesenberg, H., & Olmheim, J. (2007). Agile enterprise software development using domain-driven design and test first. *Conference on Object-Oriented Programming Systems, Languages, and Applications*. https://doi.org/10.1145/1297846.1297967

Lǐ, H. (n.d.). How to deal with dependencies in large scale? part 2: clear intent. *www.linkedin.com*. https://www.linkedin.com/pulse/how-deal-dependencies-large-scale-agile-part-2-clean-up-h%C3%A0o-l%C7%90/?trk=public\_post

Likness, J. (2012). *Building Windows 8 Apps with C# and XAML*. Addison-Wesley.

Marinescu, F., & Avram, A. (2007). *Domain-Driven Design Quickly*. Lulu.com.

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education.

Martin, R. C. (2017). *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Prentice Hall.

Millett, S., & Tune, N. (2015). *Patterns, Principles, and Practices of Domain-Driven Design*. John Wiley & Sons.

Myers, B. (2022, January 5). Red, Green, Refactor. What is Test-Driven Development? *Medium*. https://medium.com/codecastpublication/red-green-refactor-what-is-test-driven-development-302794e06c

Nguyen, P., Song, H., Chauvel, F., Muller, R., Boyar, S., & Levin, E. (2019). Using microservices for non-intrusive customization of multi-tenant SaaS. In *Foundations of Software Engineering*. https://doi.org/10.1145/3338906.3340452

Nilsson, J. (2006). *Applying Domain-Driven Design and Patterns : With Examples in C# and .NET*.

Otun, O. (n.d.). Domain-Driven Design With Event Sourcing, Akka Cluster Sharding, Cassandra, Kafka, Scala. Distributed & Reactive Trading System. FinTech Use Case. *www.linkedin.com*. https://www.linkedin.com/pulse/domain-driven-design-event-sourcing-akka-cluster-sharding-otun/

Palermo, J. (2019). *.NET DevOps for Azure: A Developer’s Guide to DevOps Architecture the Right Way*. Apress.

Rademacher, F., Sachweh, S., & Zündorf, A. (2017). Towards a UML Profile for Domain-Driven Design of Microservice Architectures. In *Lecture Notes in Computer Science* (pp. 230–245). Springer Science+Business Media. https://doi.org/10.1007/978-3-319-74781-1\_17

Rademacher, F., Sorgalla, J., & Sachweh, S. (2018). Challenges of Domain-Driven Microservice Design: A Model-Driven Perspective. *IEEE Software*, *35*(3), 36–43. https://doi.org/10.1109/ms.2018.2141028

Smith, S. (2018, October 5). *Introducing domain driven design - dogfood con 2018*. https://www.slideshare.net/ardalis/introducing-domain-driven-design-dogfood-con-2018

*SRP: The Single Responsibility Principle - Overcoded*. (n.d.). SRP: The Single Responsibility Principle - Overcoded. https://overcoded.dev/posts/BC-7

Uludağ, Ö., Hauder, M., Kleehaus, M., Schimpfle, C., & Matthes, F. (2018). Supporting Large-Scale Agile Development with Domain-Driven Design. *Lecture Notes in Business Information Processing*, 232–247. https://doi.org/10.1007/978-3-319-91602-6\_16

Vernon, V. (2013). *Implementing Domain-Driven Design*. Addison-Wesley.

Vernon, V. (2016). *Domain-Driven Design Distilled*. Addison-Wesley Professional.

Vettor, R. (2023, February 16). *Architecting Cloud Native .NET Applications for Azure*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/

Villaça, L. A., Azevedo, L. G., & Baião, F. A. (2018). Query strategies on polyglot persistence in microservices. In *ACM Symposium on Applied Computing*. https://doi.org/10.1145/3167132.3167316

Wagner, B. (2017). *More Effective C# (Includes Content Update Program): 50 Specific Ways to Improve Your C#*. Addison-Wesley Professional.

Wlaschin, S. (2018). *Domain Modeling Made Functional: Tackle Software Complexity with Domain-Driven Design and F#*. Pragmatic Bookshelf.

Zimarev, A. (2019). *Hands-On Domain-Driven Design with .NET Core: Tackling complexity in the heart of software by putting DDD principles into practice*. Packt Publishing Ltd.