Name	Studies
Adding functionality takes longer	[1]
Ambiguous Interface	[2]
API Versioning	[3], [4], [5]
Architectural erosion	[1]
Architectural/Technical Complexity	[6], [7]
Business Logic Inside Communication Layer	[8], [9], [7]
Coarse Services	[10]
Communicating the Importance of Assurance	[6]
Coordination Between Decentralized Teams	[6]
Crossing API	[11]
Cyclic Dependency	[3], [4], [2], [12], [13], [5]
Data Coupling	[11]
Dense Structure	[2]
Different Middleware Tech. for Communication	[8], [9]
Dismiss Documentation	[10]
Distributed Code Repositories	[6]
Distributed Monolith	[11]
Distributed Tracing is not supported on services and/or facades or services communicate without using a central intermediary component	[14]
Duplicate code	[1]
Endpoint-Based Service Interaction	[15], [16], [17]
Envy	[18]
ESB Misuse	[15]
ESB Usage	[3], [4]
Excessive Diversity	[19]
Excessive number of small products	[7]
Evolutionary Coupling	[11]
Feature Concentration	[2]
Forgetting About the CAP Theorem	[10]
Gluttony	[18]
God Component	[2]
Greed	[18]
Greedy Service Container	[10]
Grinding Dusty	[10]
Hard-Coded Endpoints	[3], [4], [5]
High issue resolution time	[1]
Hub-Like Dependency	[2], [12], [13]
Inadequate deployment process	[1]
Inadequate Testing	[1], [6]
Inadequate Use of APIs	[7]
Inappropriate Service Intimacy	[3], [4]
Insufficient Metadata	[19]
Insufficient metadata in the messages	[7]
Insufficient Monitoring	[5]
Integrating Legacy Code	[6]
Inter-service dependency (Ripples)	[6]
Large/complex components	[1]
Leak of Service Abstraction	[18]
Learn as You Go	[10]
Local Logging	[5]
Low release frequency	[1]
Lust	[18]

Manual Configuration	[5]
Manual handling of network issues	[7]
Mastering Technologies	[6]
Mega Service	[5]
Mega-Service	[18]
Microservice Coupling	[19], [7]
Microservice Greedy	[3], [4]
Microservices Integration	[6]
Misuse of Internal Shared Libraries	[19]
Missing/Outdated Documentation	[1], [6]
Multiple Service Instances per Host	[5]
Multiple Services in One Container	[15]
Multiple Services per Deployment Unit	[16]
Nano Service	[5], [20]
No API Gateway	[3], [4], [14], [5], [16], [17]
No Continuous Integration (CI) / Continuous Delivery (CD) (NCI)	[5]
No Health Check	[5]
No Standardized Communication Model	[8], [9], [7]
No System-Centric View	[6]
Not Having an API Gateway	[18], [21], [15]
Outdated Library	[1], [21]
Overwhelming amount of unnecessary settings	[7]
Pride	[18]
Problematic dependency	[1]
Reusing third-party implementations	[7]
Retiring Components	[11]
Rewrite All Services into Microservices at Once	[10]
Scattered Functionality	[2]
Service Chain	[20]
Service Cutting	[6]
Shared Database	[3], [4], [5], [16]
Shared Libraries	[3], [4], [5], [18], [22]
Shared Persistence	[18], [15]
Single DevOps Toolchain	[10]
Single Layer Teams	[10], [15]
Sloth	[18]
Static Contract Pitfall	[18]
Technological Heterogeneity	[6], [7]
The Knot	[20]
Thinking Microservices Are a Silver Bullet	[10]
Timeout (Dogpiles)	[18]
Timeouts	[5]
Tool/Process Frustration and Patronization	[6]
Too Many Pont-to-Point (PtP) Connections	[9], [8]
Too Many Standards	[3], [4], [18]
Unhealthy Metric Usage	[6]
Unplanned Data Sharing/Synchronization	[19], [7]
Unstable API	[11]
Unstable Dependency	[2]
Weak Source Code and Knowledge management	[8], [9]
Woobly Service Interactions	[15], [16], [17]
Wrath	[18]

Wrong Cuts	[3], [4], [18], [5]

Table 1. Microservice Smells and Their Reference Articles

References

1. Bogner, J.; Fritzsch, J.; Wagner, S.; Zimmermann, A. Limiting Technical Debt with Maintainability Assurance: An Industry Survey on Used Techniques and Differences with Service- and Microservice-Based Systems. In Proceedings of the International Conference on Technical Debt, 2018, p. 125–133. https://doi.org/10.1145/3194164.3194166.
2. Capilla, R.; Fontana, F.A.; Mikkonen, T.; Bacchiega, P.; Salamanca, V. Detecting Architecture Debt in Micro-Service Open-Source Projects. In Proceedings of the 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2023, p. 394 – 401. https://doi.org/10.1109/SEAA60479.2023.00066.
3. Pulnil, S.; Senivongse, T. A Microservices Quality Model Based on Microservices Anti-patterns. In Proceedings of the 19th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2022. https://doi.org/10.1109/JCSSE54890.2022.9836297.
4. Walker, A.; Das, D.; Cerny, T. Automated Microservice Code-Smell Detection. In Proceedings of the Information Science and Applications. Springer Singapore, 2021, p. 211–221. https://doi.org/10.1007/978-981-33-6385-4_20.
5. Tighilt, R.; Abdellatif, M.; Trabelsi, I.; Madern, L.; Moha, N.; Guéhéneuc, Y.G. On the maintenance support for microservice-based systems through the specification and the detection of microservice antipatterns. *Journal of Systems and Software* **2023**, *204*, 111755. https://doi.org/https://doi.org/10.1016/j.jss.2023.111755.
6. Bogner, J.; Fritzsch, J.; Wagner, S.; Zimmermann, A. Assuring the Evolvability of Microservices: Insights into Industry Practices and Challenges. In Proceedings of the International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 546–556. https://doi.org/10.1109/ICSME.2019.00089.
7. Toledo, S.; Martini, A.; Sjøberg, D. Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. *Journal of Systems and Software* **2021**. https://doi.org/https://doi.org/10.1016/j.jss.2021.110968.
8. Rademacher, F.; Sachweh, S.; Zundorf, A. A modeling method for systematic architecture reconstruction of microservice-based software systems. In Proceedings of the Lecture Notes in Business Information Processing, 2020, pp. 311 – 326. https://doi.org/10.1007/978-3-030-49418-6_21.
9. Toledo, S.; Martini, A.; Przybyszewska, A.; Sjøberg, D. Architectural Technical Debt in Microservices: A Case Study in a Large Company. In Proceedings of the TechDebt, 2019. https://doi.org/10.1109/TechDebt.2019.00026.
10. Carrasco, A.; Van Bladel, B.; Demeyer, S. Migrating towards microservices: Migration and architecture smells. In Proceedings of the International Workshop on Refactoring (IWoR/ASE), 2018, pp. 1 – 6.
11. Fang, H.; Cai, Y.; Kazman, R.; Lefever, J. Identifying Anti-Patterns in Distributed Systems With Heterogeneous Dependencies. In Proceedings of the 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C), 2023, pp. 116–120. https://doi.org/10.1109/ICSA-C57050.2023.00035.
12. Pigazzini, I.; Di Nucci, D.; Fontana, F.A.; Belotti, M. Exploiting dynamic analysis for architectural smell detection: a preliminary study. In Proceedings of the 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2022, p. 282 – 289. https://doi.org/10.1109/SEAA56994.2022.00051.
13. Arcelli Fontana, F.; Camilli, M.; Rendina, D.; Taraboi, A.G.; Trubiani, C. Impact of Architectural Smells on Software Performance: an Exploratory Study. In Proceedings of the Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering. Association for Computing Machinery, 2023, EASE '23, p. 22–31. https://doi.org/10.1145/3593434.3593442.
14. Ntentos, E.; Zdun, U.; Plakidas, K.; Geiger, S. Evaluating and Improving Microservice Architecture Conformance to Architectural Design Decisions. In Proceedings of the Service-Oriented Computing. Springer International Publishing, 2021, p. 188–203. https://doi.org/10.1007/978-3-030-91431-8_12.
15. Neri, D.; Soldani, J.; Zimmermann, O.; Brogi, A. Design principles, architectural smells and refactorings for microservices: a multivocal review. 2020, Vol. 35, pp. 3 – 15.
16. Soldani., J.; Marinò., M.; Brogi., A. Semi-Automated Smell Resolution in Kubernetes-Deployed Microservices. In Proceedings of the Proceedings of the 13th International Conference on Cloud Computing and Services Science - CLOSER. INSTICC, SciTePress, 2023, pp. 34–45. https://doi.org/10.5220/0011845500003488.
17. Soldani, J.; Muntoni, G.; Neri, D.; Brogi, A. The TOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures. *Software - Practice and Experience* **2021**, *51*, 1591 – 1621. https://doi.org/10.1002/spe.2974.
18. Taibi, D.; Lenarduzzi, V. On the Definition of Microservice Bad Smells. *IEEE Software* **2018**. https://doi.org/10.1109/MS.2018.2141031.
19. De Toledo, S.S.; Martini, A.; Nguyen, P.H.; Sjøberg, D.I.K. Accumulation and Prioritization of Architectural Debt in Three Companies Migrating to Microservices. *IEEE Access* **2022**, *10*, 37422–37445. https://doi.org/10.1109/ACCESS.2022.3158648.
20. Gamage, I.U.P.; Perera, I. Using dependency graph and graph theory concepts to identify anti-patterns in a microservices system: A tool-based approach. In Proceedings of the 2021 Moratuwa Engineering Research Conference (MERCon), 2021, pp. 699–704. https://doi.org/10.1109/MERCon52712.2021.9525743.

21. Lenarduzzi, V.; Lomio, F.; Saarimäki, N.; Taibi, D. Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software* **2020**, *169*, 110710. https://doi.org/https://doi.org/10.1016/j.jss.2020.110710.
22. Toledo, S.; Martini, A.; Sjøberg, D. Improving agility by managing shared libraries in microservices. *Lecture Notes in Business Information Processing (LNBIP)* **2020**. https://doi.org/10.1007/978-3-030-58858-8_20.