Category	Name	Studies
API Management	Ambiguous Interface	[1]
	API Versioning	[2] [3] [4] [5]
	Crossing API	[6]
	Inadequate Use of APIs	[7]
	No API Gateway	[2] [3] [8] [4] [9] [10] [11][12] [13]
	Unstable API	[6]
	Static Contract Pitfall	[11]
	Services are directly connected to clients	[8]
Dependency Management	Cyclic dependency	[1] [3] [2] [4] [5] [14]
	Data coupling	[6]
	Evolutionary coupling	[6]
	Hub-like dependency	[1] [5] [14]
	Inter-service dependency (ripples)	[15]
	Problematic dependency	[16]
	Reusing third-party implementations	[7]
	Service chain	[17]
	The knot	[17]
	Unstable dependency	[1]
	Sloth	[11]
Middleware	Different middleware technologies for communication	[18] [19]
	Distributed tracing is not supported on services and/or facades or services communicate without using a central intermediary component	[8]
	ESB misuse	[13]
	ESB usage	[2] [3]
	Misuse of Internal Shared Libraries	[20]
	Services communicate without using an intermediary	[8]
Discovery	Hard-coded endpoints	[2] [3] [4]
	Manual handling of network issues	[7]
	Endpoint-based service interactions	[9] [10] [13]
	Too many point-to-point (PtP) connections	[18] [19]
	Woobly service interactions	[9] [10] [9] [13]
	Timeout	[4] [11]
Data Management	Data ownership	[11]
	Shared database	[2] [3] [4] [9]
	Shared libraries	[2] [3] [4] [11] [18] [21]
	Unplanned data sharing and synchronization	[7] [20]
Decomposition	Coarse services	[22]
	Distributed monolith	[6]
	Envy	[11]
	Feature concentration	[1]
	God component	[1]

	Greedy service container	[22]
	Grinding dusty	[22]
	Large/complex components	[16]
	Mega service	[4] [11]
	Microservice coupling	[7] [20]
	Microservice greedy	[2] [3]
	Microservices integration	[15]
	Multiple services per deployment unit	[4]
	Nano service	[4] [17]
	Scattered functionality	[1]
	Service cutting	[15]
	Wrong cuts	[2] [3] [11] [4]
Team/Product Management	Communicating the importance of assurance	[15]
	Coordination between decentralized teams	[15]
	Team coupling	[6]
	Team/product greed	[11]
	Adding functionality takes longer	[16]
Architectural Standards	Architectural erosion	[16]
	Architectural/technical complexity	[7] [15]
	Business Logic Inside Communication Layer	[7] [18] [19]
	Distributed Code Repositories	[15]
	Dense structure	[1]
	Excessive diversity	[20]
	Excessive number of small products	[7]
	Leak of service abstraction	[11]
	No system-centric view	[15]
	No Standardized Communication Model	[7] [18] [19]
	Overwhelming amount of unnecessary settings	[7]
	Retiring Components	[6]
	Technological heterogeneity	[7] [15]
	Thinking microservices are a silver bullet	[22]
	Too many standards	[2], [3] [11]
	Tool/Process Frustration and Patronization	[15]
Quality Assurance	Bottleneck service	[17]
	Defects with new releases	[16]
	Duplicate code	[16]
	Gluttony	[11]
	High issue resolution time	[16]
	Inadequate testing	[15] [16]
	Inappropriate service intimacy	[2] [3]
	Insufficient metadata	[7] [20]
	Insufficient monitoring	[4]
	Local logging	[4]
	No health check	[4]
	Pride	[11]
	Woobly service interaction	[9] [10] [13]
	Wrath	[11]
	Unhealthy metric usage	[15]
DevOps (CI/CD)	Manual configuration	[4]
	Inadequate deployment process	[16]
	Manual handling of network issues	[7]
	Multiple service instances per host	[4]
	Multiple services in one container	[13]

	Multiple services per deployment unit	[9]
	No continuous integration (CI) / continuous delivery (CD) (NCI)	[4]
	Single DevOps toolchain	[22]
	Low release frequency	[16]
Documentation	Missing / outdated documentation	[15] [16] [22]
	Weak source code and knowledge management	[18] [19]
Migration	Integrating legacy code	[15]
	Learn as You Go	[22]
	Outdated library	[12] [16]
	Rewrite all services into microservices at once	[22]
	Forgetting About the CAP Theorem	[22]

Table 1. Microservice Smells and Their Reference Articles

References

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
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. 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.
9. 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.
10. 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.
11. Taibi, D.; Lenarduzzi, V. On the Definition of Microservice Bad Smells. *IEEE Software* **2018**. https://doi.org/10.1109/MS.2018.2141031.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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.