0		Goals	Phases	Granularity	Collector					Representation						Refactorization						Quality Assessment			Open Issues
					Source					Syntax			Type	Visualization		Domain Knowledge		Community Refactorization Algorithms			Functionality Refactorization	Metrics			
					Development	Runtime	Higher Level Models	Documentation	Collection Technique	Structure	Behavior	Evolution		Relations	Elements	Aggregation Criteria	Expert Knowledge	Decomposition Algorithms	Optimization Algorithms	Editing		Single Decomposition	Several Decompositions	Graphical Comparation	
				- Methods - Classes - Entities - Functionalities - Packages - OpenAPI - File	- Source code - Database Schema - Versions History	- Runtime log traces - User interactions	- Business process models - Use cases - Data flow diagrams	- Directly in de code - Paper documents - Digital format	- Static Analysis - Dynamic Analysis - Version Analysis - Model Analysis	- Inner structure - Inter structure -- Association -- Inheritance	- Call graph - Sequence of accesses - Frequency - CPU level - Memory level - Response time	- By Contributor - By Task	- Monolith - Microservices	- Structural relationship - Control flow - Data flow - ...	- Cluster - Element - Sequences	- Functionality - Lexical - Task - Contributor - Connectivity	- Business capability - Business domain	- Hierarchical Clustering - Community detection - K-means++ - Formal conceptual analysis	- Genetic algorithms	- Clusters -- Merge -- Split -- Move element -- Duplicate element	- Asynchronization - Granularity Refinement	- Coupling - Cohesion - Complexity - Elements Size - Team Size	- MoJoFM	- Color - Distance - Size	
Abgaz2023	**Decomposition of Monolith Applications into Microservices Architectures: A Systematic Review**	- RQ1 What are the primary phases of monolith to microservices decomposition and the major constituent elements of those phases? - RQ2 What are the existing approaches, tools and methods observed in the decomposition of monolith applications into microservices? - RQ3 What are the metrics, datasets, and benchmarks used for evaluating and validating monolith decomposition into microservices? - RQ4 What research gaps can be identified in the current literature?	- Phase I - Input Collection. - Phase II - Monolith Analysis. - Phase III - Microservices Identification. - Phase IV - Microservices Optimisation. - Phase V - Microservices Evaluation. - Phase VI - Microservices Deployment.	- Static Data - Dynamic Data - Model Data - Version-based Data - Methods - Functions - Classes - Packages - Software artifact (use cases, data stories) - URI	- Code-based (CIC) - Version-based (VIC)	- Log-based (LIC)	- Model-based (MIC)	- Process documentation - Interviews	- Static Analysis - Dynamic Analysis - Domain Analysis - Version Analysis	- Structure	- Call Graph - Frequency - Performance logs with method-level CPU and memory consumption [70] - Web access logs with client IP, requested resource URI, timestamp, HTTP status code, and document size, response time [75,82].	- Version history - Contributor history			- Matrix - Vector - Graph	- Classes -- Coupling -- Cohesion -- Topic Modeling -- Business object modeling - URI -- Association with at runtime	- Rules of thumb - Expert knowledge	- Hierarchical Clustering - Kmeans Clustering - Affinity Propagation - Genetic Algorithm - Fast Community Clustering - Girvan-Newman and Epidemic Label Propagation - Louvain Algorithm - Graph Matching for Ontologies - Dependency Graph Analysis and Topic Modeling - Network based Community Detection Algorithm - SAviF - Minimum Spanning Tree - Hungarian Algorithm - CO-GCN3 - Microservices optimization -- Non-dominated Sorting Genetic Algorithm II -- Strength Pareto Evolutionary Algorithm				- Coupling -- Afferent coupling -- Efferent coupling -- Structural coupling -- Instability -- Interface number - Cohesion -- Lack of cohesion -- Cohesion at message level -- Cohesion at domain level -- Relational cohesion - Modularity -- Conceptual similarity -- Structural similarity - Execution time - CPU usage - Memory usage - Network overhead - Number of classes per cluster - Number of clusters - Non-extreme distribution			- Proposes Phase VI: Microservices Deployment, though it was only identified in one study and they raise the issue on the suitability for production environments - Supervised techniques might increase the opportunity for human involvement in the decomposition loop
Abdellatif2021	**A taxonomy of service identification approaches for legacy software systems modernization**	- RQ1: What are the inputs used by SIAs? - RQ2: What are the processes followed by SIAs? - RQ3: What are the outputs of SIAs? - RQ4: What is the usability of SIAs?	- Input -- Domain artifacts -- Non executable model representations -- Executabel model representations - Process -- Quality -- Direction -- Automation -- Analysis type - Output -- Architecture -- Service types - Usability -- Validation -- Accuracy precision -- Tool support -- Results quality	- Architecture -- Services -- Microservices - Service types -- Business services -- Enterprise services -- Application services -- Entity services -- Utility services -- Infrastructure services	- Source Code - Database	- Runtime artifacts - Log Traces - User Application Interaction - Test cases	- Domain model - Business Process Model - Use Case - Activity diagram - Data Flow Diagram - State Machine Diagram	- Documentation	- Static analysis - Dynamic analysis - Lexical analysis - Historic analysis						- Concept lattices		- Human expertise	- Wrapping - Genetic algorithms - Clustering - Custom heuristics - Semi-automatic -- Formal conceptual analysis		- General guidelines		- Coupling - Cohesion - Granularity - Number of services			- They consider the direction of the process -- Top-down -- Bottom-up -- Mixed - They consider the tool support -- Industry ready -- Open source -- Prototype -- No tool support - They consider the results quality associated with the validation type -- High -- Medium -- Low - They do not distinguish between the monolith inputs and its treatment (monolith representation and domain knowledge) - They include migration do SOA - Quality requirements -- Reuse -- Maintainability -- Interoperability -- Self-containment -- Composability - Ontology
Mohottige2025	**Reengineering software systems into microservices: State-of-the-art and future directions**	- RQ1 How did research on the reengineering of software systems into microservice-based systems develop over time? - RQ2 What approaches are used to reengineer software systems into microservice-based systems, and how are reengineered systems evaluated? - RQ2.1 What classes of approaches (e.g., static and dynamic) exist? - RQ2.2 What tools exist and which level of automation do they support? - RQ2.3 Which techniques/algorithms are used? - RQ2.4 How is data (e.g., software logs) used? - RQ2.5 How are the reengineered systems evaluated? - RQ3 What are the challenges and limitations of existing methods for reengineering software systems into microservice-based systems?	- Microservice Identification - Modeling - Microservice Extraction - Microservice Evaluation	- Classes - Methods - Packages - Database entities	- Source Code - Database schemata - Source code repository histories	- Software logs - Server events logs - Runtime monitoring	- Software desing artifacts -- UML -- Data flow diagrams -- Architectural documents -- Use cases -- User stories -- Ubiquitous language - Domain Models - Domain-driven design pattern - Task-driven design patter		- Static Analysis - Dynamic Analysis - Artifact-Driven - Hybrid Analysis	- Structural inheritance - Structural interaction						- Matrix/table based extraction - Neural networks - Graph-based extraction - Machine learning & genetic algorithms		- Expert analysis		- Cohesion - Relational Cohesion - Cohesion at message & domain level - Cohesion at method level / internal cohesion - Lack of Cohesion - Density - Modularity - Structural Modularity - Conceptual Modularity - Mean cluster factor - Feature Modularization - Runtime Performance - Latency / execution time - Throughput, accuracy and performance - Scalability - Availability - Efficiency gain - Network overhead - Independence - Interface number - Interactions number / Inter partition call percentage - Operation number - Internal & external co-change frequency - Number of duplicated classes - Frequency of external calls - Coupling - Coupling between microservices - Structural Coupling - Afferent & Efferent Coupling - Internal & External Coupling - Coupling at method level - Absolute importance & dependence - Microservice Interdependence - Instability Index	- Quality of the decomposition - DB transactional purity - Business Context Purity - Non-Extreme Distribution / classes per microservice - Precision, Recall & F-Measure - Code redundancy rate - Domain redundancy rate - Team size reduction - Reuse - Estimated development time - MoJoFM - MoJoSim		- Modeling Techniques:
Saucedo2025	**Migration of monolithic systems to microservices: A systematic mapping study**	- Q1: How is monolithic systems to microservices migration done? - Sub-Q1: Which processes and techniques have been used to migrate towards microservices? - Sub-Q2: Which tools have been used in the migration from monolithic systems to microservices? - Q2: Which factors have caused the migration of monolithic systems to microservices? - Q3: What are the issues and benefits of migrating towards microservices?	1. Planning 2. Analysis 3. Design 4. Execution 5. Monotoring	- Business Function - Data Entity - Dataflow Diagram - Entity - Nanoentity - Class - Method - Package - Table - API - URI	- Source code (static) - Test cases - databases - stored procedures - tables - repositories and Git history of commits and files.	- OpenAPI specifications - API - Hardware resource dependencies - Log traces - Hardware utilization metrics - Web Access logs - Performance logs		- Requirements specifications - Use cases - Use cases specifications - Design documents - entity-relationship design models - feature list - business processes - requirements models - model operations and data in TOSCA language - Domain Specific Language (DSL) architecture description - XMI system models	- static - dynamic - documentation based - source control based - data based								- Domain or architectural expert	- Word Embeddings - Word2Vec - CodeBERT - Code2Vec - Label Propagation - SVM - Clustering - Affinity Propagation - Density - Distribution - Graph - Hierarchical - Partitioning - Topic Detection - Rule-based - Neural Networks	- Genetic Algorithms			- Internal Quality Attributes - Coupling - Cohesion - Database Quality - Evolvability - Microservices Quality - Redundancy - Size - External Quality Attributes - Availability - Performance - Resource usage - Scalability - Organizational Metrics - Cost - Complexity - Team - External Evaluation			Self-supervised, Semi-supervised, supervised and unsupervised
Capuano2022	**A Systematic Literature Review on Migration to Microservices: a Quality Attributes perspective**	- RQ1: Which studies implement a quality-driven approach to migrate to microservices? - RQ2: Which are the quality attributes analyzed in the migration phases? - RQ3: In which migration phase the quality-driven process is implemented?	- comprehension of the system (CP) - identifying the microservices (MIP)		- Code - Dependencies - Classes - Methods - Packages - Files - Directories	- Logs - Traces - Use case testing - User experience testing - User stories testing - Data	- Business process	- Requirements	- Static analysis - Dynamic analysis - Data-driven analysis - Domain analysis											- Cohesion - Coupling - Functionality - Scalability			

0		Goals	Phases	Granularity	Collector					Representation						Refactorization						Quality Assessment			Open Issues
					Source				Collection Technique	Syntax			Type	Visualization		Domain Knowledge		Community Refactorization Algorithms		Editing	Functionality Refactorization	Metrics		Graphical Comparation	
					Development	Runtime	Higher Level Models	Documentation		Structure	Behavior	Evolution		Relations	Elements	Aggregation Criteria	Expert Knowledge	Decomposition Algorithms	Optimization Algorithms			Single Decomposition	Several Decompositions		
				- Methods - Classes - Entities - Functionalities - Packages - OpenAPI - File	- Source code - Database Schema - Versions History	- Runtime log traces - User interactions	- Business process models - Use cases - Data flow diagrams	- Directly in de code - Paper documents - Digital format	- Static Analysis - Dynamic Analysis - Version Analysis - Model Analysis	- Inner structure - Inter structure -- Association -- Inheritance	- Call graph - Sequence of accesses - Frequency - CPU level - Memory level - Response time	- By Contributor - By Task	- Monolith - Microservices	- Structural relationship - Control flow - Data flow - ...	- Cluster - Element - Sequences	- Functionality - Lexical - Task - Contributor - Connectivity	- Business capability - Business domain	- Hierarchical Clustering - Community detection - K-means++ - Formal conceptual analysis	- Genetic algorithms	- Clusters -- Merge -- Split -- Move element -- Duplicate element	- Asynchronization - Granularity Refinement	- Coupling - Cohesion - Complexity - Elements Size - Team Size	- MoJoFM	- Color - Distance - Size	
Oumoussa2024	**Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review**	- RQ1: How has the area of study on microservices identification evolved ? - RQ2: What is the current state-of-the-art in microservices identification research? - RQ3: What are the current and potential challenges associated with microservices identification?			- Codebase				- Model Driven - Static Analysis / Static Code Analysis (SCA) - Dynamic Analysis / Dynamic Microservices Composition (DMC) - Meta-Data Assistance (MDA) - Workload-Data Assistance (WDA)					- Structure of the monolithic system		- Functionality - Dependencies		- Automation Algorithms - Clustering-based - Dependency-based - Data-Driven	- Genetic Algorithms - Neural Networks			- Coupling - Interface Number (IFN) - Lack of Cohesion (LOC) - Structural Modularity (SMQ) - Execution Time - Memory Usage - Cohesion Metrics - Performance Metrics - CPU usage - Network Overhead - Precision and Recall - F1-Score - Changes in microservices (e.g., addition, modification, removal) - Changes in lines of code - Code Smells - Microservices popularity		- Manual Comparation	
Auer2021	**From monolithic systems to Microservices: An assessment framework**	RQ1. Why did companies migrate to Microservices? RQ2. Which information/metrics was/were collected before and after the migration? RQ3. Which information/metrics was/were considered useful by the practitioners?																				Performance - Response time - CPU utilization - Path length - Usage of containers - Waiting time - Impact of programming language Scalability - Number of requests per minute or second - Number of features per Microservice Availability - Downtime - Mean time to recover - Mean time to failure Maintenance - Complexity - Testability Process-related benefits - Development independence between teams - Continuous delivery - Reusability Personnel Cost - Development costs Infrastructure Cost - Cost per hour - Cost per million requests			
Nassima2025	**Semantic Approaches to microservice identification: A Systematic Literature Review**	RQ1: Semantic techniques - Which semantic techniques and technologies are used for microservice identification in monolithic systems? RQ2: Evaluation Methodologies - How semantic approaches to microservice identification evaluated and validated? RQ3: Challenges and Limitations - What are the main challenges, limitations, and barriers to the adoption of semantic approaches? RQ4: Future Research Directions - What are the most promising future research directions for semantic approaches to microservice identification?			- Source code											- Semantic Analysis -- NLP Techniques -- Ontology Techniques -- Hybrid -- LLM-like techniques		- Semantic Analysis -- NLP Techniques -- Ontology Techniques -- Hybrid -- LLM-like techniques							They identify the approaches but do not describe how they are applied.
Oumoussa2025	**The Ontology-Based Mapping of Microservice Identification Approaches: A Systematic Study of Migration Strategies from Monolithic to Microservice Architectures**	RQ1. How can challenges encountered during the microservice identification phase of monolith-to-microservice migration be classified and addressed systematically? RQ2. How can existing microservice identification strategies be categorized, and what are their performance and effectiveness metrics? RQ3. What are the typical inputs, data models, algorithms, and performance metrics used in microservice identification strategies, and how are they grouped? RQ4. What levels of automation exist in microservice identification approaches, and how are they distributed across different methodologies? RQ5. What trends and emerging research directions are evident in microservice identification, and how can they be visualized in a systematic mapping framework?			- Technical - Source code - API specifications - Database schemas	- Technical - Execution logs - Runtime traces	Business oriented - Process models - User stories - Functional requirements - BPMN diagrams - Domain expertise	Hybrid - Combined technical–business - Stakeholder inputs - System context								Graph based - Semantic clustering - Knowledge graphs - Neural networks Relational - Data flow matrices - Component matrices - Dependency analysis AI-based - Deep learning - NLP models - Evolutionary algorithms		Graph based - Semantic clustering - Knowledge graphs - Neural networks Relational - Data flow matrices - Component matrices - Dependency analysis AI-based - Deep learning - NLP models - Evolutionary algorithms				Functional - Cohesion - Coupling - Modularity	Performance - Response time - Resource usage - Scalability Business value - ROI metrics - Time to market - Customer satisfaction		It does not decompose the technique into their elements: representation, knowledge and algorithms
Trabelsi2025	**A Systematic Literature Review of Machine Learning Approaches for Migrating Monolithic Systems to Microservices**	- RQ1: Which migration phases are automated by ML? - RQ2: How are ML migration inputs characterised in terms of type, granularity, source, and preprocessing? - RQ3: What ML techniques and learning paradigms are applied in migration approaches? - RQ4: How are ML-based migration approaches evaluated with respect to metrics, benchmarks, tools, and success criteria? - RQ5: What challenges arise in using ML approaches?	(migration phases, not identification phases) 1. Pre-Migration 2. Identification 3. Packaging 4. Deployment 5. Monitoring	- Application level - Use case - Files - URI - API - System level - Services Microservices - Node - Component - Performance Metric - Process level - Operation - Activity - Code level - Class - Method - Data level - Table - Entity	- Source code - Configuration files - Data files	- Resource metrics - CPU - Memory usage - Monitoring metrics - Trace Logs - Workloads	- Business processes - UML diagrams - User stories	- API Documentation - QoS Contraints - > Expert knowledge - Architectural Recomendations - > Expert knowledge - Functionality Descriptions - Technical Artifacts	- Data extraction - Data cleaning - Noise reduction - Normalization - Code analysis - static analysis - Semantic analysis - Structural analysis - Dependency modeling - Vector - Graph - Matrix	Structural features - Class dependencies - Method calls - Data dependencies - Call graph dependencies	Behavioral features - Invocation paths - Log events - Contextual log entries - Temporal patterns						- Classical machine learning algorithms - Naive Bayes - Support Vector Machines - clustering algorithm - Graph-based methods	- Genetic algorithms - Reinforcement learning			- System Behavior - Response time - Resource utilization - Energy consumption - SLA violation rate - Software Design - Coupling - Cohesion - Size and Scope - Modularity (structural modularity) - Complexity (cognitive complexity)	- Prediction - Precision - Recall - F1-score - Accuracy - MCC - Clustering - Dunn Index - Silhouette Score - Newman-Girvan Modularity - Non-extremity distribution - Maximum Cluster Size - > Expert knowledge - Number of Singletons Clusters - > Expert knowledge - Developer - Devoper Validation - > Expert knowledge and editing - Closeness to Manual Expert Analysis - > Expert knowledge and editing - > Several Decompositions		- Learning Paradigms - Selected Features - Aggragation criteria over representation but at the same time it's not

0		Goals	Phases	Granularity	Collector					Representation						Refactorization						Quality Assessment			Open Issues
					Source				Collection Technique	Syntax			Type	Visualization		Domain Knowledge		Community Refactorization			Functionality Refactorization	Metrics			
																		Algorithms		Editing				Graphical Comparation	
					Development	Runtime	Higher Level Models	Documentation		Structure	Behavior	Evolution		Relations	Elements	Aggregation Criteria	Expert Knowledge	Decomposition Algorithms	Optimization Algorithms			Single Decomposition	Several Decompositions		
				- Methods - Classes - Entities - Functionalities - Packages - OpenAPI - File	- Source code - Database Schema - Versions History	- Runtime log traces - User interactions	- Business process models - Use cases - Data flow diagrams	- Directly in de code - Paper documents - Digital format	- Static Analysis - Dynamic Analysis - Version Analysis - Model Analysis	- Inner structure - Inter structure -- Association -- Inheritance	- Call graph - Sequence of accesses - Frequency - CPU level - Memory level - Response time	- By Contributor - By Task	- Monolith - Microservices	- Structural relationship - Control flow - Data flow - ...	- Cluster - Element - Sequences	- Functionality - Lexical - Task - Contributor - Connectivity	- Business capability - Business domain	- Hierarchical Clustering - Community detection - K-means++ - Formal conceptual analysis	- Genetic algorithms	- Clusters -- Merge -- Split -- Move element -- Duplicate element	- Asynchronization - Granularity Refinement	- Coupling - Cohesion - Complexity - Elements Size - Team Size	- MoJoFM	- Color - Distance - Size	
Yang2025	**Full stack optimization of microservice architecture: systematic review and research opportunity**	- What is the principle of microservice decomposition? What are the existing methods to support microservice decomposition decisions? How are these methods implemented, and what is the general flow? What are the advantages and disadvantages? - What is the principle of microservice placement? What existing methods are used in the microservice placement process, and how are they implemented? What are the advantages and disadvantages of these methods? What factors should be considered in the placement process? - What is the principle of resource scheduling in microservices? What is the goal? What strategies are used during microservice scheduling, and how are they implemented? What are the advantages and disadvantages of these strategies? What are the characteristics of these resource scheduling methods? - What is the principle of microservice migration? What are the differences between migrating microservices in different environments? What factors should be considered during the migration process? What strategies are used during microservice migration, and how are they implemented? What is the goal? What are the advantages and disadvantages of these strategies? - What is the principle of microservice anomaly detection, and what processes are involved? What monitoring tools are used in anomaly detection, and what data is monitored? What methods are used in the microservice anomaly detection process, what is the rationale, what progress has been made, and what are the advantages and disadvantages?		- Fine-grained - Coarse-grained - API	- Source code - OpeaAPI specification - Commit history	- Runtime traces - Access logs	- Business Process Model - Business logic model - UML diagrams - Interface description - Non-functional requirements	- Prior knowledge - Domain information	- Static analysis - Dynamic analysis	- Structural representation	- Dynamic representation - Call graph - System dependency graph					- Semantic similarity - Structure similarity	- Expert knowledge	- Clustering algorithms - Hierarchical clustering (HCA) - Collaborative clustering (CCA) - Breadth-first search (BFS) with predefined rules	- Machine learning techniques - Genetic algorithm						