THE BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

DATABASE SYSTEMS JOURNAL

SPECIAL ISSUE: EMERGING CONCEPTS AND TRENDS IN COMPUTING

2026

LISTED IN

RePEc, EBSCO, DOAJ, Open J-Gate,
Cabell's Directories of Publishing Opportunities,
Index Copernicus, Google Scholar,
Directory of Science, Cite Factor,
Electronic Journals Library

BIG DATA
NoSQL
ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
BUSINESS INTELLIGENCE
IoT
CLOUD COMPUTING
BLOCKCHAIN
DATA WAREHOUSES
DATABASES

ISSN: 2069 – 3230
dbjournal.ro

Database Systems Journal BOARD

Contact
CaleaDorobanţilor, no. 15-17, room 2017, Bucharest, Romania
Web: https://dbjournal.ro/
E-mail: editordbjournal@gmail.com

CONTENTS

SPECIAL ISSUE

When Concepts Become Mantras: How Are Computing Terms Turning into Technological Slogans

Daniela TUDORICĂ[1], Antonia PĂTRAȘCU[23], Bogdan George TUDORICĂ[4*]
[1] Petroleum-Gas University of Ploiești, Computer Science, Information Technology, Mathematics, and Physics department
[2] University of Craiova, "Eugeniu Carada" Doctoral School of Economic Sciences
[3] Petroleum-Gas University of Ploiești, Cybernetics, Economic Informatics, Finance, and Accounting department
[4] Petroleum-Gas University of Ploiești, Cybernetics, Economic Informatics, Finance, and Accounting department
dtudorica@upg-ploiesti.ro, antonia.patrascu@upg-ploiesti.ro, btudorica@upg-ploiesti.ro

Computer science is a technical field, yet its language does not evolve only through the linear accumulation of concepts, methods, and tools. Periodically, certain expressions become dominant: data warehouse, cloud first, big data, digital transformation, blockchain, digital twin, generative AI, and AI agents. Some denote real technical solutions, with durable effects on professional practice. Others operate mainly as signals of topicality, formulas through which projects, products, and strategies align themselves with the expectations of a given period.

This article examines that phenomenon through an analysis of computer science terms that have circulated intensely, sometimes to the point of exhaustion. It follows their period of maximum visibility, initial technical meaning, dominant promise, the mechanism through which the term became fashionable, its gradual detachment from technical content, the indicators of semantic emptying, and what remained after the terminological wave passed. Particular attention is given to the effects of such fashions: excessive focus, unjustified spending, projects started through institutional imitation, architectures adopted without necessity, and the relabeling of older activities so that they appear compatible with the discourse of the moment.

The article's thesis is that fashionable terms should not be rejected wholesale. They become problematic only when they cease to require a description of the mechanism. A mature technical concept allows concrete questions: what problem it solves, what architecture it presupposes, what data it uses, what risks it introduces, and how the result is verified. A slogan, by contrast, promises modernization without indicating the means through which it is produced.

Keywords: *computer science, technology fashions, technical language, hype, data warehouse, cloud computing, big data, digital transformation, artificial intelligence.*

1 Introduction

Every professional field has its fashions. Computer science is no exception. Although it is tied to mathematics, engineering, formal logic, and technical infrastructures, it also produces a vocabulary with social, managerial, and commercial value. Certain expressions circulate not only because they are useful but also because they signal belonging to the present. A project that invokes "big data," "cloud," "AI," "digital twin," or "agentic AI" appears, on first reading, more current than one that speaks of databases, distributed applications, statistical models, or the automation of processes.

This difference in perception matters. Fashionable terms influence project funding, research topics, institutional procurement, conference titles, university course content, and the way vendors present their products. They can concentrate attention on real problems. They can create professional communities and accelerate the standardization of practices. At the same

time, they can produce imitation, disproportionate expenditure, and projects designed more to check a term than to solve a problem.

This ambivalence is important. Data warehouse was not a mere fashion. The concept responded to a real need: separating operational systems from analytical systems, integrating historical data, and constructing a coherent basis for reporting and decision-making [1], [2]. Yet the term was often used for any large database, any shared file repository, or any somewhat ambitious reporting system. In its technical form, it required discussions about sources, granularity, transformations, data quality, history, dimensional models, and access rights. In its worn-out form, it required only a label.

A similar mechanism can be observed in the expression cloud first. As an IT policy principle, it can have a reasonable meaning: before building or expanding local infrastructure, the organization should evaluate cloud services. In dogmatic form, however, the expression suggests that cloud migration is implicitly superior, regardless of costs, regulation, latency, data sovereignty, vendor dependence, or available competencies. The motto ceases to be an analytical criterion and becomes an order.

In recent years, the same dynamic has shifted toward artificial intelligence. Formulas such as "AI-driven," "AI-native," "copilot," "agentic AI," and "AI agents" appear in products, strategies, research projects, and institutional documents. Sometimes they describe real functions: large language models, generation systems, semantic search, automatic classification, tool orchestration, or contextual assistance. At other times, they conceal a modest mechanism: a heuristic rule, a decorative conversational interface, a better-packaged search engine, or a classical automation renamed for the present.

The central problem is not the novelty of terms. A living field needs new concepts. The problem appears when a term becomes independent of its mechanism. At that point, vocabulary no longer describes technical reality; it replaces it. The project no longer needs to show how it works. It only needs to declare that it belongs to a prestigious family of solutions.

The article proposes a critical reading of this phenomenon. It does not aim to polemically dismantle new technologies but to separate technical content from semantic inflation. The working question is simple: What remains of a term after the fashion passes?

2 From technical concept to prestige formula

A technical term becomes influential when it solves a problem recognized by a professional community. In its initial phase, its use is relatively strict. Specialists use it to describe an architecture, a method, a practice, or a class of tools. Its meaning is tied to verifiable mechanisms.

Over time, the term moves from the technical space into managerial and commercial space. Vendors include it in presentations. Consultants introduce it into reports. Universities add it to courses. Funding programs use it in calls. Organizations adopt it in strategies. At this stage, the word begins to circulate faster than its explanation.

The process is not necessarily negative. The extension of a term can help establish a common language. For example, cloud computing allowed clearer discussions of infrastructure as a service, managed platforms, subscription-delivered applications, and resource elasticity [3], [4]. DevOps named a real problem: the rupture between software development and the operation of systems in production. Data science gave visibility to a combination of competences that had previously existed separately: statistics, programming, data analysis, and domain knowledge.

Degeneration appears when the term comes to be used without constraints. "Cloud" comes to mean any system accessible online. "Big data" designates any dataset that exceeds the comfort of a conventional table. "Digital transformation" covers anything from the digitization of a form to the complete reorganization of an operating

model. "Digital twin" is applied to 3D models, dashboards, static simulations, or interactive maps, even when the operational link with the physical system is absent. "AI" becomes attached to applications that contain neither machine learning, nor inference, nor generative models, but only predefined rules. This moment can be called semantic detachment. The term continues to circulate but no longer requires a description of the architecture. It no longer obliges demonstration. It no longer demands evaluation criteria. Through its mere appearance, it promises competence, modernity, or alignment with a dominant trend.

A clear sign of this detachment is the appearance of broad attributive formulas: "intelligent platform", "digital ecosystem", "AI-driven solution", "cloud-native architecture", "blockchain-based system", and "agentic component." These formulas can be correct, but only if followed by precise explanations. Without them, they become rhetorical markers.

3 The life cycle of a terminological fashion

Terminological fashion has a relatively stable cycle. Not all concepts pass through exactly the same stages, and their duration differs from case to case, but the general succession through which a fashionable term passes can usually be described in eight steps.

The real problem - At the beginning there is a technical or organizational difficulty. Organizations cannot integrate data. Applications are hard to deliver. Servers are underused. Industrial systems are insufficiently monitored. Machine learning models cannot easily be moved into production. Users need fast access to information in documents. The problem is concrete.

The technical concept - A solution or family of solutions appears. An architecture, method, or practice is proposed: data warehouse, SOA, cloud, containers, DevOps, MLOps, RAG. At this stage, the term is still tied to mechanisms. To use it correctly, one must be able to describe what it does and how it works.

Professional consecration - Technical communities adopt the term. Books, conferences, products, courses, certifications, and best practices appear. Some implementations succeed. The term acquires prestige because it seems to solve a problem felt by many actors.

Managerial extension - The term includes strategies, institutional documents, project calls, reports, and commercial presentations. At this stage, the distance grows between those who know the mechanism and those who use the term to formulate objectives. Much confusion begins here: the word is taken over faster than the technical discipline it presupposes.

Semantic inflation - The field of application broadens. The term covers increasingly different cases. Some are legitimate, others forced. In this phase, any product can become a "platform," any automation can become "AI," any visual model can become a "digital twin," any API can be presented as a "microservice," and any virtual space can be called a "metaverse."

Independence from mechanism - The term functions as a password. It produces the desired effect without describing the system's operation. Instead of explaining how data are collected, cleaned, what model is used, and how the result is evaluated, the document says that the solution is "AI-driven." Instead of justifying decentralization, it says that the system uses blockchain. Instead of explaining the physical model, synchronization, and simulation, it says that a digital twin will be created.

Wear - After excessive use, the term becomes suspect. Specialists begin to avoid it or replace it with more precise formulations. Jokes, criticisms, warnings, and "washing" formulas appear: cloud washing, AI washing, blockchain washing, agent washing. Organizations that have spent heavily without results become more cautious. The term does not disappear immediately, but it loses its seductive force.

Sedimentation - After the fashion passes, the technically valid part remains. The data warehouse remains as an analytical architecture. The cloud remains as a model for delivering infrastructure and services. Containers remain in software delivery chains. DevOps remains as a set of operational practices. Generative models remain as tools for text, code, images, and conversational interfaces. The rest withdraws into the archives of technological rhetoric.

4 Families of technological promises

Fashionable terms do not appear in isolation. They group themselves into families of promises. This classification is useful because it shows that computer science periodically returns to the same aspirations: automating software development, extracting value from data, hiding infrastructure, securing access, decentralizing control, reducing consumption, automating processes, digitally doubling the physical world, and automating cognition.

The promise of automating software development

This family includes 4GL, CASE tools, RAD, low-code/no-code, platform engineering, AI coding assistants, and, in part, prompt engineering. The promise is the same: applications will be created faster, with less code, by people closer to the business problem and less dependent on specialized programmers.

The results are mixed. Some tools have reduced effort for interfaces, reports, prototypes, departmental applications, and simple automations. They have not, however, eliminated design, testing, integration, security, error handling, and maintenance. In many cases, complexity has not disappeared; it has moved into configurations, connectors, hidden rules, and platform dependencies.

The indicator of semantic emptying appears when "no code," "internal platform," or "AI-assisted development" are presented as equivalents of development without architecture, governance, or technical responsibility.

The promise of organizational knowledge

This family includes MIS, DSS, knowledge management, data warehouse, business intelligence, data mining, big data, data science, data lake, lakehouse, data mesh, RAG, and AI analytics. The general formula is that the organization will make better decisions if it organizes its data and turns it into information or knowledge.

This promise has produced many real results. Without databases, reporting, analytical models, indicators, and data-quality processes, large organizations cannot function intelligibly. Yet it has also produced large, expensive, and sometimes useless projects. Data warehouses have been built without clear analytical questions, dashboards without real decision-makers, data lakes without governance, data meshes without domain owners, and predictive models without integration into decision processes.

The indicator of semantic emptying appears when one speaks of "the value of data" without specifying who uses the data, for what decision, at what level of quality, and with what operational consequence.

The promise of elastic and invisible infrastructure

This family includes mainframe, time-sharing, grid computing, client-server, SOA, cloud computing, cloud first, serverless, microservices, containers, Kubernetes, edge computing, FinOps, and, in part, observability. It promises that infrastructure will become more flexible, more scalable, easier to manage, or better correlated with economic value.

Each wave solved part of the problems of its period. The mainframe provided stability and centralization. Client-server brought distribution and better interfaces. SOA attempted to integrate heterogeneous systems. The cloud provided elasticity and rapid access to managed services. Serverless promised a reduction in infrastructure administration. Containers improved

portability. FinOps emerged as a reaction to cloud costs that had become hard to control.
The risk appears when architecture becomes dogma. A small application does not necessarily need microservices. A system with regulated data should not automatically be moved to the public cloud. A stable service does not become better merely because it is containerized. A serverless function does not eliminate the architectural problem; it moves it toward events, execution time, state, costs, and observability.
The indicator of semantic emptying is the sentence "We will adopt architecture X for modernization," without analysis of constraints, without total cost of ownership, and without justification of the fit between problem and architecture.

The promise of security through new principles
This family includes Zero Trust, privacy-enhancing technologies, confidential computing, some forms of identity management, and, in certain contexts, post-quantum cryptography. The dominant promise is that access, privacy, and trust can be reconstructed for distributed, hybrid, and exposed environments.
Zero Trust does not mean generalized distrust as an administrative attitude, but continuous verification, contextual access control, segmentation, and reduction of implicit trust [5]. PETs do not mean magical anonymization, but a family of techniques through which data can be used with reduced exposure: differential privacy, secure multiparty computation, homomorphic encryption, synthetic data, and protected execution environments [6], [7], [8].
The indicator of semantic emptying appears when security is expressed through labels rather than threat models. If a project declares Zero Trust without access policies, an inventory of identities, segmentation, and monitoring, the term is decorative. If a project invokes PETs without specifying what privacy risk is reduced and by what technique, the promise remains verbal.
The promise of decentralization
This family includes peer-to-peer, grid computing in part of its history, blockchain, DLT, smart contracts, Web3, decentralized identity, and tokenization. The dominant promise is the reduction of dependence on a central authority through protocols, cryptography, consensus, or the distribution of responsibility.
There are situations in which this promise makes sense. A distributed ledger can be useful when actors do not trust one another, when auditability is important, and when there is no authority accepted by all participants. The problem appears when blockchain is introduced into systems where there is already a legitimate central institution, a conventional database would suffice, and the cost of distributed consensus is not justified.
The indicator of semantic emptying appears when "blockchain-based," "Web3," or "decentralized" are used without explaining the trust problem.

The promise of digitally doubling the physical world
This family includes IoT, smart city, Industry 4.0, cyber-physical systems, edge computing, Green IT, and digital twin. The terms indicate the attempt to connect objects, machines, buildings, cities, and industrial processes to digital systems that collect data, analyze system state, and permit intervention.
The valid technical part is important. Industrial monitoring, predictive maintenance, process control, energy consumption analysis, and simulation of physical systems can produce measurable benefits. Yet this terminological family is vulnerable to simplification. A city does not become "smart" by installing isolated sensors. A factory does not become "Industry 4.0" merely by acquiring robots. A 3D model is not automatically a digital twin. A data center does not become "green" through a declaration of intent.
The indicator of semantic emptying appears when representation replaces function and when the vocabulary of sustainability is not

tied to energy measurements, emissions, hardware use, or the life cycle of equipment.

The promise of process automation

This family includes workflow automation, BPM, RPA, hyperautomation, process mining, low-code/no-code, and some forms of AI agents. The promise is that repetitive, fragmented, or human-intervention-dependent processes can be executed faster, cheaper, and more uniformly through software.

RPA was attractive precisely because it could automate legacy systems without modifying them. The software robot imitates user actions in existing interfaces. Hyperautomation extended this promise by combining RPA with AI, machine learning, process mining, OCR, and workflow orchestration.

The indicator of semantic emptying appears when automation is proposed before the process is understood. If the process is unstable, ambiguous, or full of exceptions, automating it will produce errors faster, not better results.

The promise of cognitive automation

This family brings together expert systems, knowledge-based systems, machine learning, deep learning, generative AI, LLMs, copilots, RAG, AI agents, agentic AI, and, at a more speculative distance, quantum AI. The promise is the automation of activities considered cognitive: diagnosis, classification, prediction, writing, programming, summarization, search, planning, dialogue, and task execution.

This is one of the strongest families of terms, because it directly touches the relationship between intellectual work and automation. In its serious form, it involves data, models, training, evaluation, errors, bias, security, integration, and human control. In its worn-out form, the term "AI" becomes synonymous with "modern" or "automatic."

The indicator of semantic emptying appears when the system is described as "intelligent" without specifying what model it uses, what data it processes, what decision it influences, how results are evaluated, and who is responsible for errors.

The promise of reforming a sector through technology

This family includes FinTech, EdTech, HealthTech, GovTech, InsurTech, LegalTech, and similar terms. They do not designate a single technology but a sectoral repositioning: finance through technology, education through technology, health through technology, and administration through technology.

FinTech is the most visible case. The term can designate mobile payments, digital banking, alternative lending, robo-advisory, open banking, regtech, insurtech, cryptocurrencies, or payment infrastructures. The promise is efficiency, broader access to services, a better user experience, and competition with traditional financial institutions [9].

The indicator of semantic emptying appears when the suffix "Tech" suggests innovation without indicating the technology, the economic model, the regulatory framework, and the risk for users.

5 Families of technological promises

Time-sharing

Period of maximum visibility: the 1960s and 1970s.

Initial meaning: time-sharing designated the shared use of a computing system by multiple users, usually through terminals connected to a central computer. Resources were expensive, and their use had to be shared. The idea was simple: processor time could be distributed among interactive sessions so that multiple users could experience direct access to the system.

Dominant promise: broader access to computing. The computer was no longer only an isolated machine reserved for batch processing, but an instrument with which the user could interact.

How it became fashionable: the term became associated with the democratization of computing in universities, laboratories, and large organizations. A rhetoric of access

formed around it: more users, more applications, more flexibility.
How it became independent of mechanism: in administrative discourse, time-sharing sometimes came to mean any form of remote access or shared computer use, without explanations of process scheduling, terminals, interactive sessions, or shared resources.
Indicator of semantic emptying: expressions such as "shared computing services" without specification of the resources actually shared and the access model.
Possible effects of the fashion: the emphasis on access could conceal the real limits of the systems: variable response time, dependence on the central computer, difficult administration, and high costs. Still, in this case, the fashion was closely tied to an authentic technical need.
What remained: the interactive session, access to shared resources, the separation between user and infrastructure, and the idea of a computing service delivered by a central system. Retrospectively, time-sharing anticipates part of the logic of cloud services, although the technical means were entirely different.

Mainframe computing
Period of maximum visibility: the 1960s through the 1980s, with persistence in critical domains to the present.
Initial meaning: the term designated computing performed on large, centralized, highly reliable systems used by banks, administrations, large companies, universities, and research centers.
Dominant promise: control, stability, and processing capacity for large organizations. The mainframe was associated with the seriousness of computing infrastructure.
How it became fashionable: in many institutions, computerization was identified with the acquisition of a powerful central system. The large computer became a sign of organizational modernization.
How it became independent of mechanism: "mainframe" sometimes came to designate any central infrastructure, regardless of its real architecture. The term also became a cultural label: old system, hard to modify, but critical.
Indicator of semantic emptying: using the term for any legacy or centralized system, without distinguishing among architecture, age, languages, transactional workloads, and availability requirements.
Possible effects of the fashion: excessive centralization could lead to vendor dependence, high costs, and operational rigidity. At the same time, later criticism of the mainframe was often unfair. Many such systems remained in production because they were stable, not because organizations ignored novelty.
What remained: critical transactional systems, the culture of reliability, centralized resource administration, and the idea that some computing tasks require continuity more than novelty.

Management Information Systems / MIS
Period of maximum visibility: the 1970s and 1980s.
Initial meaning: MIS designated information systems intended to collect, process, and present information needed by management. The emphasis was on reports, indicators, summaries, and decision support.
Dominant promise: rational management based on information. Managers were to receive relevant data, on time, in a form that allowed organizational control.
How it became fashionable: the term entered organizational and university language. It seemed to offer a bridge between computer science and management at a time when the computer was becoming an administrative instrument.
How it became independent of mechanism: MIS came to designate almost any computer application in an organization, even if it did not provide managerial information in the proper sense.
Indicator of semantic emptying: the use of the expression "management information system" for ordinary operational applications, without indicators, reports, aggregations, or a decision-support role.

Possible effects of the fashion: some organizations confused data collection with decision support. Having reports does not automatically mean making better decisions. Reports can become administrative rituals if they are not tied to actions, responsibilities, and evaluation criteria.
What remained: the distinction between operational data and managerial information, the idea of periodic reporting, dashboards, and decision support.

Database and the relational model
Period of maximum visibility: the 1970s through the 1990s, with structural persistence to the present.
Initial meaning: a database is a designated, organized collection of data administered by a specialized system. The relational model introduced a powerful idea: data can be represented through relations, while users should be protected from the details of physical storage [10].
Dominant promise: order, consistency, and flexible querying. Data no longer had to be scattered through files that were difficult to control.
How it became fashionable: almost any serious application began to be conceived around a database. The term became part of the common language of computerization.
How it became independent of mechanism: "database" came to designate any list of data: Excel files, document archives, isolated tables, and unvalidated collections. In common usage, the difference between a database, a tabular file, and an informal collection became blurred.
Indicator of semantic emptying: expressions such as "we have a database" followed by the description of an unsecured file, without a data model, integrity rules, or administration.
Possible effects of the fashion: in some projects, the mere existence of a "database" was confused with the existence of a data strategy. Isolated, redundant, poorly documented databases appeared and later became difficult to integrate.
What remained: DBMSs, SQL, data modeling, transactions, referential integrity, normalization, and the separation between logical structure and physical storage. This is an example of a term that outlived fashion and became conceptual infrastructure.

Expert systems
Period of maximum visibility: the 1980s.
Initial meaning: expert systems were applications based on rules, knowledge bases, and inference mechanisms, designed to reproduce in part the reasoning of a specialist in a narrow domain.
Dominant promise: automated availability of expertise. The knowledge of a specialist could be captured, formalized, and used by a program.
How it became fashionable: the term succeeded because it promised practical artificial intelligence. In companies, laboratories, and public institutions, the expert system seemed a pragmatic way to automate specialized decisions.
How it became independent of mechanism: "expert" was attached to applications that offered only menus, simple recommendations, or very transparent rules. In some cases, the term suggested intelligence, although the system merely applied predefined conditions.
Indicator of semantic emptying: the expression "expert system" used without an explicit knowledge base, without an inference engine, and without a mechanism for explaining conclusions.
Possible effects of the fashion: some projects underestimated the difficulty of extracting knowledge from experts. Rules were hard to maintain, conflicts among rules appeared frequently, and systems became fragile outside anticipated cases.
What remained: rule engines, decision-support systems, configurators, formal knowledge representation, and the idea of explaining a decision. Part of today's vocabulary around explainable AI indirectly revisits questions already raised by expert systems.

Knowledge-based systems and knowledge management

Period of maximum visibility: the 1980s through the 2000s.
Initial meaning: knowledge-based systems designated systems that used explicitly represented knowledge. Knowledge management extended the discussion toward capturing, organizing, and sharing knowledge within an organization.
Dominant promise: the organization would not lose people's experience and would reuse accumulated knowledge. Useful information would no longer remain in the heads of a few experts or in isolated documents.
How it became fashionable: the term was taken over by consulting and management. It became attractive because it promised to turn informal experience into a controllable organizational resource.
How it became independent of mechanism: knowledge management sometimes came to mean the mere existence of an intranet, a document archive, or an unmaintained wiki.
Indicator of semantic emptying: expressions such as "knowledge management platform" without taxonomy, editorial responsibilities, update flows, version control, and retrieval mechanisms.
Possible effects of the fashion: many organizations invested in document repositories that were populated at first and then abandoned. Knowledge does not circulate merely because an instrument exists. Time, incentives, editorial discipline, and trust are necessary.
What remained: knowledge bases, internal documentation, technical wikis, procedures, question-and-answer systems, communities of practice, and concern for preserving organizational memory.

CASE tools
Period of maximum visibility: the late 1980s and the 1990s.
Initial meaning: CASE, Computer-Aided Software Engineering, designated tools that assisted analysis, design, modeling, and sometimes code generation. They attempted to introduce engineering discipline into software development.
Dominant promise: more controlled, standardized software development, less dependent on improvisation. Diagrams, specifications, and partial code generation were expected to reduce errors.
How it became fashionable: CASE appeared in a context in which software project failure was already a major theme. CASE tools seemed a natural reaction to that crisis.
How it became independent of mechanism: in some organizations, adopting a CASE tool was confused with the maturity of the software process. The diagram became a sign of method even when analysis was weak.
Indicator of semantic emptying: the existence of extensive visual models, difficult to maintain, that do not actually guide implementation and are not updated after the code changes.
Possible effects of the fashion: investments in heavy tools used only formally; documentation produced for audit rather than design; teams forced to over-model applications that first needed clearer requirements.
What remained: visual modeling, generation of code skeletons, modern IDEs, UML tools, requirements traceability, and the idea that software design can be assisted by tools, but not replaced by them.

4GL and the promise of programming closer to the user
Period of maximum visibility: the 1980s and 1990s.
Initial meaning: fourth-generation languages were more declarative or more oriented toward business applications than traditional procedural languages. They allowed reports, queries, screens, and operations on data to be expressed more rapidly.
Dominant promise: applications developed faster, with less code and greater involvement of business users.
How it became fashionable: the term was associated with reducing dependence on programmers. In organizations, this promise was attractive because the backlog of applications was large and classical development seemed slow.

How it became independent of mechanism: 4GL was applied broadly to any tool perceived as easier than conventional programming, even when the technical differences were significant.
Indicator of semantic emptying: presenting a tool as a "fourth-generation language" without explaining the execution model, the type of applications targeted, and its limits.
Possible effects of the fashion: quick applications that were hard to maintain; vendor dependence; business logic hidden in forms, reports, and configurations. The same tension appears today in some low-code platforms.
What remained: SQL, report generators, declarative development, visual tools for business applications, and the legitimate idea that not every problem must be solved through low-level procedural code.

Client-server
Period of maximum visibility: the 1990s.
Initial meaning: the client-server architecture separated the interface and part of the application logic, located on the user's computer, from central services such as the database or the application server.
Dominant promise: better interfaces, distribution of tasks, improved performance, and autonomy from centralized systems.
How it became fashionable: with the spread of local networks and personal computers, client-server became the natural model for modern organizational applications. It was presented as an alternative to mainframes and "dumb" terminals.
How it became independent of mechanism: almost any networked application began to be called client-server, even when the distribution of responsibilities was not clearly analyzed.
Indicator of semantic emptying: the use of the term without specifying roles: what the client executes, what the server executes, where the data are, where rules are validated, and how concurrency is handled.
Possible effects of the fashion: applications with logic scattered across tiers, difficult installations on user workstations, versions hard to synchronize, and dependence on fragile local networks. Some organizations discovered that distribution brings flexibility but also administrative costs.
What remained: separation of responsibilities, database servers, application servers, distributed applications, and the basic vocabulary of communication among software components.

Object-oriented programming
Period of maximum visibility: the 1990s and early 2000s, with a stable presence to the present.
Initial meaning: object-oriented programming organizes programs through classes and objects, using encapsulation, inheritance, polymorphism, and the binding of data to the operations that use them.
Dominant promise: software that is more modular, more reusable, and easier to extend. Object modeling seemed to bring programming closer to the real world.
How it became fashionable: languages such as C++, Java, and C# consolidated object-oriented programming as a dominant model in industry and education. "Object-oriented" became almost synonymous with modern programming.
How it became independent of mechanism: the term was invoked for code that used classes merely as procedural groupings. In other cases, inheritance was overused, and class hierarchies became more important than the problem solved.
Indicator of semantic emptying: projects that declare an "object-oriented architecture" but contain classes with confused responsibilities, anemic objects, artificial hierarchies, or static methods that reproduce procedural programming.
Possible effects of the fashion: over-engineering, useless classes, patterns applied mechanically, rigid hierarchies, and code that is hard to follow. In education, premature emphasis on OOP could obscure elementary algorithmic thinking.
What remained: encapsulation, interfaces, polymorphism, design patterns, component architecture, and a useful mental model for

many types of applications. Object-oriented programming survived fashion because it has real mechanisms, even if it was sometimes abused.

RAD / Rapid Application Development
Period of maximum visibility: the 1990s.
Initial meaning: RAD designated rapid application development through prototyping, visual tools, short iterations, and user involvement.
Dominant promise: applications delivered faster, with fewer long specification cycles and a better response to user requirements.
How it became fashionable: organizations were dissatisfied with slow, rigid, expensive projects. RAD seemed an escape from excessive formalism.
How it became independent of mechanism: "rapid" sometimes became synonymous with "insufficiently analyzed." Visual tools were confused with the method.
Indicator of semantic emptying: the promise of a "rapidly developed" application without controlled prototypes, validation with users, and acceptance criteria.
Possible effects of the fashion: prototypes prematurely turned into production applications, fragile architectures, accumulation of technical compromises, and insufficient documentation.
What remained: prototyping, iterative development, early validation with users, visually built interfaces, and the idea that a software project must learn from the beneficiary's reaction.

Data warehouse
Period of maximum visibility: the 1990s and 2000s, with continuing developments.
Initial meaning: a data warehouse is an analytical infrastructure for integrating data from operational systems, preserving history, and supporting reporting and analysis. In the classical sense, it presupposes the separation between transactional and analytical processing, extraction and transformation processes, data models, and quality control [1], [2].
Dominant promise: a coherent view of the organization. Decision-makers would have access to clean, integrated, comparable data.
How it became fashionable: the term was adopted in computerization projects, consulting, and business intelligence. It became almost mandatory in organizations that wanted serious managerial reporting.
How it became independent of mechanism: the data warehouse began to be confused with any large database, any reporting server, or any periodically exported data archive.
Indicator of semantic emptying: "we will build a data warehouse" without specification of sources, granularity, history, data quality, transformation rules, dimensional model, and final users.
Possible effects of the fashion: large, costly projects blocked by source integration; warehouses fed with poor-quality data; reports that reproduce the inconsistencies of operational systems. A data warehouse built without clear analytical questions can become an expensive repository of ambiguities.
What remained: dimensional modeling, ETL/ELT, data governance, history preservation, separation of analysis from transactions, and the discipline of reporting on controlled data.

OLAP
Period of maximum visibility: the 1990s and 2000s.
Initial meaning: OLAP, Online Analytical Processing, designated multidimensional data analysis through cubes, aggregations, hierarchies, and operations such as slicing, dicing, drill-down, and roll-up.
Dominant promise: rapid exploration of business data by analysts and managers, without manually written SQL queries.
How it became fashionable: OLAP became the natural pair of the data warehouse. If the organization had integrated data, it had to be able to analyze sales, costs, customers, products, and regions quickly.
How it became independent of mechanism: the term came to designate almost any

interactive report or any analytical tool with filters.
Indicator of semantic emptying: calling a page with charts "OLAP" without cubes, dimensions, measures, hierarchies, or controlled aggregations.
Possible effects of the fashion: investments in analysis tools without an adequate data model; rapid reports on incoherent data; the false impression that visualization solves data quality.
What remained: multidimensional analysis, analytical cubes, predefined aggregations, dimension hierarchies, and exploratory interaction with data.

Data mining
Period of maximum visibility: the 1990s and 2000s, later partially absorbed into data science and machine learning.
Initial meaning: data mining designated the discovery of patterns, relationships, association rules, groupings, or predictive models in datasets. The term was linked to concrete algorithms: classification, clustering, decision trees, association rules, anomaly detection.
Dominant promise: data contain hidden knowledge that can be extracted automatically.
How it became fashionable: the term caught on in business because it suggested a simple and seductive analogy: data are a mine, and algorithms extract value.
How it became independent of mechanism: any analysis more advanced than descriptive reporting began to be called data mining. Sometimes the term was even used for elementary statistics.
Indicator of semantic emptying: "we apply data mining" without specifying the algorithm, variables, training set, validation criterion, and interpretation of results.
Possible effects of the fashion: blind search for patterns, correlations without causal meaning, overfitting, fragile commercial conclusions, and projects in which the algorithm was chosen before the research question.
What remained: classification, clustering, association rules, anomaly detection, predictive analysis, and the vocabulary that later fed data science.

E-business, e-commerce, e-learning, e-government
Period of maximum visibility: the late 1990s and the 2000s.
Initial meaning: the prefix "e-" indicated the transfer of activities into the electronic environment, especially through the Internet. Commerce, education, administration, and services could be performed or supported through digital platforms.
Dominant promise: faster processes, broader access, lower costs, and remotely available services.
How it became fashionable: after the Web expanded, the prefix "e-" became a marker of modernity. Almost any activity could be redescribed as an electronic variant.
How it became independent of mechanism: the mere existence of a website, form, or online account was enough to call a service "e-." Transformation of the real process could be absent.
Indicator of semantic emptying: "e-government" reduced to downloading a PDF form that must be printed, signed, and physically submitted.
Possible effects of the fashion: superficial digitization, duplication of paper workflows with online interfaces, and increased bureaucracy instead of its reduction. In education, e-learning was sometimes confused with uploading files to a platform.
What remained: online commerce, educational platforms, digital public services, electronic signatures, online payments, digital forms, and remote access.

Portal
Period of maximum visibility: the 2000s.
Initial meaning: the portal was a single point of access to information, applications, and services. In the organizational environment, it integrated internal resources, authentication, documents, news, applications, and possibly reports.

Dominant promise: unified access. The user no longer had to search for resources in several systems.
How it became fashionable: institutions, companies, and universities began to present their websites as portals. The term suggested more structure and value than "website."
How it became independent of mechanism: any larger site, with several menus, could be called a portal, even if it integrated no services and did not personalize access.
Indicator of semantic emptying: "portal" used for a static site with informational pages, without authentication, services, or integration.
Possible effects of the fashion: oversized web projects, complicated menus, overloaded interfaces, and confusion between publishing information and delivering services.
What remained: single sign-on, service aggregation, intranets, institutional dashboards, and the idea of a central access point.

Dot-com and the "new economy"
Period of maximum visibility: the second half of the 1990s and the early 2000s.
Initial meaning: dot-com designated companies whose activity was centered on the Internet, usually with commercial domains ending in .com. The "new economy" suggested that the Internet was changing the economic rules of growth, distribution, and value.
Dominant promise: rapid growth, global markets, low marginal costs, and business models radically different from traditional ones.
How it became fashionable: venture capital, stock listings, and the economic press amplified the term. A company seemed more valuable if it was associated with the Internet, even before it had profit, a stable model, or sufficient customers.
How it became independent of mechanism: mere online presence became, for a time, an economic argument. Questions about revenue, logistics costs, customer retention, and model sustainability were sometimes postponed.
Indicator of semantic emptying: business plans in which "Internet-based" replaces the explanation of how revenue is obtained and how costs are controlled.
Possible effects of the fashion: overinvestment, speculative valuations, companies without solid economic models, and rapid collapses. The dot-com bubble shows that a technology can be real and important even if many investments made in its name are irrational.
What remained: e-commerce, online advertising, digital platforms, web infrastructure, SaaS models, and the platform economy. The Internet was not a fashion; the illusion that any online business is automatically valuable was the fashion.

XML everywhere
Period of maximum visibility: the 2000s.
Initial meaning: XML was an extensible format for representing structured data and documents. It was textual, standardized, and suitable for data exchange among heterogeneous systems.
Dominant promise: universal interoperability. If systems could exchange XML, then they could communicate more easily.
How it became fashionable: XML was adopted in standards, web services, configurations, documents, data flows, and enterprise applications. It became an almost default answer to the problem of data exchange.
How it became independent of mechanism: XML was also used in situations where it was too verbose, hard to read, or unnecessarily complicated. Sometimes XML structures reproduced the disorder of source data, only in a more standardized form.
Indicator of semantic emptying: "We use XML for interoperability" without a schema, validation, a contract between systems, and controlled versions.
Possible effects of the fashion: large files, poor performance, standards hard to implement, and formal but fragile integration. A common format does not guarantee common meaning.

What remained: document formats, exchange standards, validation through schemas, SOAP in enterprise systems, configuration files, and an important lesson: common syntax does not solve semantics by itself.

Semantic Web
Period of maximum visibility: the 2000s and early 2010s.
Initial meaning: Semantic Web designated the extension of the Web from documents readable by humans to data that can be interpreted and correlated automatically by applications. The idea presupposed metadata, ontologies, identifiers, explicit relations among resources, and standards such as RDF, OWL, and SPARQL [11], [12], [13].
Dominant promise: the Web would become more intelligible to machines. Applications could automatically combine data from different sources, reducing ambiguity and dependence on manual interpretation.
How it became fashionable: The term was associated with the vision of a more intelligent Web, capable of supporting automatic reasoning, knowledge integration, and better-connected digital services.
How it became independent of mechanism: Semantic Web was sometimes invoked for any site with metadata, any controlled vocabulary, or any minimal structuring of content, without ontologies, linked data, or real semantic querying.
Indicator of semantic emptying: "semantic platform" without an explicit conceptual model, persistent identifiers, formal vocabulary, and the possibility of linking data with other sources.
Possible effects of the fashion: academic or institutional projects with elaborate ontologies but little use; models too abstract for current needs; high costs of semantic modeling; adoption difficulties outside specialized communities.
What remained: linked data, knowledge graphs, domain ontologies, better metadata, semantic querying, and part of the conceptual infrastructure used today in search systems, data integration, and AI applications.

SOA / Service-Oriented Architecture
Period of maximum visibility: the 2000s.
Initial meaning: SOA designated an architecture in which application functionality is exposed as interoperable services, with clear contracts, messaging, and the possibility of orchestration.
Dominant promise: flexible integration among heterogeneous systems. The organization could build new processes by combining existing services.
How it became fashionable: SOA was promoted intensely in enterprise architecture. Middleware vendors, consultants, and IT departments saw in it a solution for integrating old and new applications.
How it became independent of mechanism: the term was applied to any application that exposed web services. Sometimes SOA meant only the existence of a SOAP layer over unchanged systems.
Indicator of semantic emptying: enterprise diagrams with many services, but without precise contracts, governance, versioning, monitoring, and responsibility for availability.
Possible effects of the fashion: heavy infrastructures, oversized ESBs, costly integration projects, services that were too general or too granular, and dependence on complex platforms.
What remained: APIs, contracts among systems, reusable services, enterprise integration, messaging, and the idea that large applications must communicate through well-defined interfaces.

Grid computing
Period of maximum visibility: the 2000s.
Initial meaning: grid computing designated the coordinated use of distributed computing resources, located in different organizations or places, to solve problems requiring high processing power or storage.
Dominant promise: access to distributed computing power without each organization having to own the entire necessary infrastructure locally.
How it became fashionable: scientific research, simulations, bioinformatics,

particle physics, and other computationally intensive domains fed interest in grids. The term seemed to offer a global infrastructure for collaborative computing.
How it became independent of mechanism: grid computing was invoked for any distribution of tasks or any form of resource sharing, even when middleware, scheduling, federated authentication, and distributed resource administration were absent.
Indicator of semantic emptying: "grid" used for a local cluster or a few connected servers, without distributed resource management and without collaboration across administrative domains.
Possible effects of the fashion: infrastructures difficult to administer, complex configurations, dependence on specialized middleware, and usability problems for researchers who needed applications rather than grid administration.
What remained: distributed computing, resource federation, infrastructures for research, large-scale job scheduling, and some ideas later absorbed by cloud and managed high-performance computing.

ERP and the promise of total integration
Period of maximum visibility: the 2000s, with continuous use.
Initial meaning: ERP, Enterprise Resource Planning, designated integrated systems for managing organizational resources: accounting, inventory, procurement, production, sales, human resources, and other processes.
Dominant promise: complete integration of the organization into a common system. Data would be entered once and used coherently across all modules.
How it became fashionable: ERP was presented as a sign of organizational maturity. For many companies, adopting an ERP became almost mandatory as processes grew in complexity.
How it became independent of mechanism: the term sometimes came to designate any large administrative application, even if there was no real integration among modules.
Indicator of semantic emptying: "ERP" used for a collection of weakly connected applications, with manual exports and redundant data.
Possible effects of the fashion: expensive projects, excessive customization, organizational resistance, dependence on consultants, and changing processes merely to fit the software. An ERP implemented without cleaning up processes can computerize disorder.
What remained: process integration, traceability, financial-accounting modules, unified management, and operational data discipline.

CRM
Period of maximum visibility: the 2000s, with continuous use.
Initial meaning: CRM, Customer Relationship Management, designated systems and practices for managing relationships with customers: contacts, opportunities, sales, support, campaigns, and interaction history.
Dominant promise: the organization would know its customers better, sell more efficiently, and provide better services.
How it became fashionable: CRM was widely adopted in sales and marketing. The term promised a shift from isolated transactions to systematically managed relationships.
How it became independent of mechanism: in some organizations, CRM came to mean a customer list, a contact book, or an application into which agents entered data for managerial control.
Indicator of semantic emptying: "CRM" without a defined sales process, responsibilities, data quality, and use of interaction history in decisions.
Possible effects of the fashion: data entry without value for direct users, bureaucratic surveillance of sales agents, many reports, and few actions.
What remained: sales pipelines, interaction history, customer segmentation, post-sales support, and integration of communication with commercial activity.

Web 2.0
Period of maximum visibility: 2004-2010.
Initial meaning: Web 2.0 designated a phase of the Web characterized by user participation, user-generated content, interactive applications, social platforms, blogs, wikis, comments, tagging, and services that exploited network effects [14].
Dominant promise: the user was no longer only a reader but a participant. The Web would become social, collaborative, and dynamic.
How it became fashionable: the term was rapidly taken over by industry, the press, and conferences. Applications that allowed profiles, comments, ratings, sharing, or communities were presented as part of the new Web.
How it became independent of mechanism: Web 2.0 came to designate any site with a more modern look or minor interactive functions. Sometimes the term described only a visual change, not a change in the user's role.
Indicator of semantic emptying: formulas such as "Web 2.0 platform" without real mechanisms of participation, moderation, contribution, collaboration, or reuse of content.
Possible effects of the fashion: overestimating online communities, building platforms without a critical mass of users, initially generated content later abandoned, and social interfaces artificially added to services that did not need them.
What remained: social platforms, wikis, user-generated content, dynamic interfaces, online collaboration models, and the idea that the value of a digital service can also be produced by the user community.

AJAX and dynamic web applications
Period of maximum visibility: 2005-2015.
Initial meaning: AJAX, Asynchronous JavaScript and XML, designated the use of JavaScript for asynchronous communication with the server, so that the page could be updated without a full reload.
Dominant promise: smoother, faster web applications, closer to desktop applications.
How it became fashionable: after the success of interactive web applications, AJAX became a sign of technical modernity. The web interface no longer had to be a rigid succession of fully loaded pages.
How it became independent of mechanism: the term began to be used for any more dynamic interface, even when XML was not used or when the mechanisms were broader than the initial technique.
Indicator of semantic emptying: "AJAX site" used for an interface that has only a few visual effects, without real asynchronous data exchange.
Possible effects of the fashion: overloaded interfaces, excessive dependence on JavaScript, accessibility difficulties, poor indexing, and errors difficult to trace. A useful technique was sometimes turned into an ornament.
What remained: interactive web applications, asynchronous communication, single-page interfaces, partial page updates, and a user experience closer to locally installed applications.

Green IT
Period of maximum visibility: 2007-present, with returns linked to cloud, data centers, AI, and environmental regulation.
Initial meaning: Green IT designated the design, use, and administration of information technologies while reducing energy consumption, emissions, electronic waste, and environmental impact.
Dominant promise: technology can be used with a lower ecological cost through energy efficiency, virtualization, server consolidation, recycling, extending the life of equipment, and more efficient software design.
How it became fashionable: the growth of data centers, pressure on energy consumption, and discourse on sustainability made Green IT an attractive term in strategies and reports.
How it became independent of mechanism: the term was used for symbolic initiatives,

such as reducing paper or purchasing equipment labeled efficient, without measuring actual consumption or life-cycle impact.
Indicator of semantic emptying: "Green IT strategy" without measured energy consumption, emissions indicators, hardware-use policies, e-waste management, and software-efficiency analysis.
Possible effects of the fashion: technological greenwashing, declarative reports, apparently ecological purchases unrelated to actual use, and cloud migration presented as green without analyzing the energy mix and effective consumption.
What remained: energy efficiency, virtualization, consolidation, efficient cooling of data centers, environmental reporting, carbon-footprint calculation, and interest in less wasteful software.

Cloud computing
Period of maximum visibility: 2006-2015, with continuing centrality.
Initial meaning: Cloud computing designates access to configurable computing resources over a network, from a shared pool of resources that can be rapidly provisioned and released [3], [4].
Dominant promise: elasticity, reduction of upfront investments, rapid access to infrastructure, managed services, and scaling on demand.
How it became fashionable: Major providers turned the cloud into a dominant economic and operational model. For many organizations, purchasing local servers began to seem outdated.
How it became independent of mechanism: "cloud" came to mean any application accessible online. Sometimes the term did not indicate elasticity, managed services, or dynamic allocation, but only external hosting.
Indicator of semantic emptying: "cloud solution" used for an application installed on a vendor's server, without scaling, automation, a clear responsibility model, or verifiable operational benefits.
Possible effects of the fashion: migration without total-cost analysis, recurring bills higher than previous infrastructure, vendor lock-in, compliance problems, misconfigurations, and loss of internal competencies. In many organizations, the cloud reduced delivery time but created hard-to-control costs where FinOps practices were absent.
What remained: IaaS, PaaS, SaaS, elastic infrastructure, managed services, automation, backup, geographic availability, data platforms, and rapid application development.

Cloud first
Period of maximum visibility: 2010-2018.
Initial meaning: "Cloud" first meant that the organization should evaluate cloud options before investing in local infrastructure. The term made sense as a reaction against the inertia of private data centers and the underuse of resources.
Dominant promise: rapid modernization, lower initial costs, and avoidance of local infrastructure that is difficult to administer.
How it became fashionable: public policies and corporate strategies adopted the formula. It was easy to communicate and seemed to give IT departments a clear direction.
How it became independent of mechanism: the motto was sometimes turned into an absolute rule: if a cloud variant exists, it must be chosen. Fit analysis was replaced by compliance.
Indicator of semantic emptying: "cloud first" used without comparison among cloud, on-premises, and hybrid, without estimating recurring costs, and without analyzing latency, security, data sovereignty, and vendor dependence.
Possible effects of the fashion: migration of unsuitable applications, incomplete re-architecting, high operational costs, audit difficulties, and vulnerabilities resulting from misconfigurations. Some projects moved to the cloud with problems that should have been solved in the application, process, or data model.

What remained: serious evaluation of cloud as an option, hybrid architectures, cloud governance policies, FinOps, and a more mature attitude: cloud when the advantage is demonstrable.

Big data
Period of maximum visibility: 2010-2018.
Initial meaning: big data designated the processing of data characterized by volume, velocity, and variety, to which other properties such as veracity and value were later added. The term was linked to distributed infrastructures, semi-structured or unstructured data, and large-scale analysis.
Dominant promise: massive data contain hidden economic, social, or scientific value that can be extracted through new technologies.
How it became fashionable: the growth of digital data, social networks, logs, sensors, and web platforms made the term attractive. Companies began to use it to signal analytical maturity.
How it became independent of mechanism: big data came to designate any dataset larger than what could comfortably be handled in a spreadsheet. Sometimes the term was invoked without real volume, fast flows, or technical variety.
Indicator of semantic emptying: "big data" projects with a few CSV files, without distributed processing, without a scaling problem, and without analytical questions that justify the technology.
Possible effects of the fashion: purchases of expensive platforms, unused clusters, uncontrolled data lakes, excessive data collection, and confusion between accumulating data and understanding them. Many data can amplify disorder if model, governance, and question are missing.
What remained: data engineering, distributed processing, data lake architectures, log analysis, streaming, cloud data platforms, and attention to semi-structured or unstructured data.

Hadoop and its ecosystem
Period of maximum visibility: 2010-2020.
Initial meaning: Hadoop was an open-source ecosystem for distributed storage and processing, associated with HDFS, MapReduce, and related tools. It responded to the problem of processing large volumes of data on clusters of ordinary machines.
Dominant promise: large-scale analytics with relatively inexpensive infrastructure.
How it became fashionable: Hadoop became the almost emblematic infrastructure of big data. Many organizations considered that a modern data strategy had to include a Hadoop cluster.
How it became independent of mechanism: installing the platform was sometimes treated as a result in itself. The existence of the cluster was confused with the existence of an analytical capability.
Indicator of semantic emptying: "we have Hadoop" without use cases, data pipelines, competent teams, security policies, and indicators concerning the value of analysis.
Possible effects of the fashion: clusters difficult to administer, data loaded without cataloging, poor performance for certain queries, underestimated operational costs, and later migration to managed cloud services. Hadoop showed that open source does not mean free in operation.
What remained: distributed processing, scalable storage, separation between storage and processing, the ecosystem of tools for large data, and part of the vocabulary absorbed by modern cloud platforms.

Data science
Period of maximum visibility: 2012-present.
Initial meaning: data science designated the combination of statistics, programming, modeling, visualization, data engineering, and domain knowledge. The term attempted to name a hybrid professional profile.
Dominant promise: better decisions and products through the systematic use of data.
How it became fashionable: the title data scientist became very attractive, associated with high salaries, innovation, and strategic positions in organizations. Universities, companies, and educational platforms built programs around the term.

How it became independent of mechanism: data science came to cover very different roles: BI analyst, statistician, data engineer, ML researcher, Python programmer, dashboard specialist. Sometimes any analysis in Python or any notebook was called data science.
Indicator of semantic emptying: "data science project" without a hypothesis, without a data model, without evaluation, without a link to a decision, and without reproducibility.
Possible effects of the fashion: confused hiring, unbalanced educational programs, pilot projects that do not reach production, and excessive emphasis on models at the expense of data. In many organizations, the problem was not the lack of algorithms, but the lack of clean data and processes that would use the results.
What remained: reproducible notebooks, hybrid professional profiles, predictive modeling, visualization, data as an operational resource, and the move from descriptive reporting to explanatory and predictive analysis.

Machine learning
Period of maximum visibility: 2012-present, with much older roots.
Initial meaning: machine learning designates methods through which a system builds models from data rather than relying exclusively on manually written rules. It includes classification, regression, clustering, recommendation, anomaly detection, and other families of techniques.
Dominant promise: prediction and automation based on data. Systems can identify patterns that are hard to formalize manually.
How it became fashionable: successes in image recognition, natural language processing, recommendations, online advertising, and commercial predictions brought the term to the foreground. Deep learning amplified its visibility.
How it became independent of mechanism: machine learning began to be used for solutions that apply fixed rules, manual scores, or descriptive statistics. The term was sometimes used as a synonym for "algorithm."
Indicator of semantic emptying: "we use machine learning" without a training set, explanatory variables, validation method, metrics, error, monitoring, and comparison with a baseline model.
Possible effects of the fashion: models applied where simple rules were sufficient, overfitting, opaque decisions, bias, results impossible to explain to beneficiaries, and projects that ignore maintenance costs. A model can work in an experiment and fail in production if the distribution of data changes.
What remained: predictive models, classifiers, recommendation systems, anomaly detection, statistical evaluation, MLOps, and the integration of algorithms into real applications.

DevOps
Period of maximum visibility: 2014-present.
Initial meaning: DevOps designated the rapprochement between software development and IT operations through automation, collaboration, continuous integration, continuous delivery, monitoring, and shared responsibility for systems in production.
Dominant promise: faster and more reliable software delivery.
How it became fashionable: organizations needed to release more often, reduce deployment errors, and shorten the distance between code and production. The term was rapidly taken over by vendors, courses, and infrastructure teams.
How it became independent of mechanism: DevOps was sometimes reduced to a list of tools: Jenkins, Docker, Kubernetes, Terraform, and GitLab CI. In other cases, it became a job title, although the initial idea concerned collaboration and shared responsibility.
Indicator of semantic emptying: "we have DevOps" when there is only a separate team that administers pipelines, without changes in the way developers and administrators work.

Possible effects of the fashion: loading teams with new tools, automating poorly defined processes, fragile pipelines, and the impression that installing a CI/CD platform solves organizational culture.
What remained: continuous integration, continuous delivery, infrastructure as code, monitoring, automation, observability, and reducing the distance between development and operations.

Microservices
Period of maximum visibility: 2014-2022, with continuing use.
Initial meaning: microservices are applications composed of small, autonomous services that communicate through well-defined interfaces and can be developed, tested, delivered, and scaled independently.
Dominant promise: better scaling, team autonomy, rapid delivery, and reduction of monolith rigidity.
How it became fashionable: the success of large technology companies turned microservices into an aspirational architecture. Many organizations wanted to reproduce architectures used at scale even when they did not have the same complexity or traffic.
How it became independent of mechanism: applications were split into services without coherent domain boundaries, without observability, without stable contracts, and without adequate operational infrastructure.
Indicator of semantic emptying: "microservices architecture" in which services share the same database, are delivered simultaneously, block one another, and are managed by the same team as a distributed monolith.
Possible effects of the fashion: increased latency, difficult debugging, operational complexity, distributed transactions, high observability costs, and fragmentation of an application that did not need fragmentation. Projects can become slower in the very name of rapid delivery.
What remained: domain-based decomposition, clear APIs, autonomous services where justified, selective scaling, and the operational discipline of distributed systems.

Containers and Kubernetes
Period of maximum visibility: 2014-present.
Initial meaning: containerization isolates the application and its dependencies in a portable environment. Kubernetes orchestrates containers, managing deployment, scaling, availability, and configuration in a cluster.
Dominant promise: portability, reproducibility, and standardized administration of applications.
How it became fashionable: Docker simplified the use of containers, and Kubernetes became the dominant platform for orchestration. The terms entered the cloud-native vocabulary quickly.
How it became independent of mechanism: the container was presented as a universal deployment solution, and Kubernetes was sometimes adopted for applications that could have been managed more simply.
Indicator of semantic emptying: "cloud-native" used for an application placed in a container, but without externalized configuration, real scaling, observability, fault tolerance, or a deployment pipeline.
Possible effects of the fashion: oversized infrastructure, high administration costs, unnecessary complexity for small applications, lack of internal competences, and superficial security. Kubernetes solves real problems, but introduces an operational layer that must be justified.
What remained: reproducible images, portable deployment, orchestration, standardization of runtime environments, CI/CD integration, and a mature infrastructure for distributed applications.

Serverless
Period of maximum visibility: 2016-present.
Initial meaning: serverless designates a cloud model in which the provider administers the infrastructure, resource allocation, scaling, and part of operations, while the developer focuses on functions, events, or managed services. The name is literally inaccurate:

servers exist, but they are not directly administered by the application team.
Dominant promise: less infrastructure administration, automatic scaling, and costs correlated with actual execution.
How it became fashionable: serverless appeared as a natural continuation of the cloud. After organizations externalized infrastructure, the next step was externalizing an even larger part of runtime administration.
How it became independent of mechanism: the term was used for any managed cloud service or for applications in which servers were only contractually, not architecturally, hidden.
Indicator of semantic emptying: "serverless" without an event model, automatic scaling, latency analysis, observability, and estimation of costs by execution volume.
Possible effects of the fashion: fragmentation of logic into functions hard to follow, cold starts, difficulties of local testing, vendor lock-in, unpredictable costs at high traffic, and security problems specific to events and fine-grained permissions.
What remained: FaaS, event-driven architectures, managed services, automatic scaling, and reduced infrastructure administration for applications suited to this model.

Digital transformation
Period of maximum visibility: 2015-present.
Initial meaning: digital transformation designated the change of processes, services, operating models, and relationships with users through digital technologies. The term presupposed changing the way work is done, not merely acquiring software.
Dominant promise: an organization that is more efficient, more flexible, more data-oriented, and better adapted to the digital environment.
How it became fashionable: the term entered strategies, grants, consulting, government programs, and corporate documents. It was broad enough to include almost any IT project.
How it became independent of the mechanism: digital transformation began to designate the digitization of forms, the purchase of a system, the development of a portal, or the movement of documents into a platform, without changing processes.
Indicator of semantic emptying: "digital transformation project" without process analysis, indicators, organizational change, data integration, and involved users.
Possible effects of the fashion: broad, costly projects with small results; purchasing platforms before clarifying processes; digitizing bureaucracy instead of reducing it; organizational resistance. McKinsey's frequently cited observation that many transformations fail is useful as a warning: transformation is not an automatic result of technology [15].
What remained: digital services, workflow automation, data integration, online interaction with beneficiaries, electronic signatures, more traceable processes, and a more mature understanding that technology changes the organization only if the process changes.

Industry 4.0
Period of maximum visibility: 2015-present.
Initial meaning: Industry 4.0 designates the integration of cyber-physical systems, industrial IoT, automation, data, and advanced analytics in production.
Dominant promise: smart factories, flexible processes, continuous monitoring, predictive maintenance, and adaptive production.
How it became fashionable: the term was supported by industrial policies, funding programs, equipment vendors, and technology consulting.
How it became independent of mechanism: Industry 4.0 was used for any industrial modernization: the acquisition of sensors, robots, CNC machines, or monitoring software, even when integration was missing.
Indicator of semantic emptying: "Industry 4.0 line" without data flow, interoperability, analysis, connection to operational decisions, and data-based maintenance.
Possible effects of the fashion: investments in equipment without integration, incompatible platforms, islands of

automation, and pilot projects that do not reach industrial scale.
What remained: industrial IoT, process monitoring, predictive maintenance, integration of equipment with information systems, and production data analysis.

Smart city
Period of maximum visibility: 2015-present.
Initial meaning: A smart city designates the use of digital technologies and data to better administer urban services: mobility, energy, lighting, safety, environment, administration, and the relationship with citizens.
Dominant promise: a more efficient, connected, and better administered city.
How it became fashionable: the term was highly attractive to administrations, technology vendors, and funding programs. It allowed the city to be presented as a space of innovation.
How it became independent of mechanism: A smart city was frequently reduced to video cameras, LED lighting, mobile applications, isolated sensors, or platforms for reporting problems.
Indicator of semantic emptying: "smart city" without integrated data, governance, urban indicators, interoperability, and measurable effects on public services.
Possible effects of the fashion: visible but fragmentary projects; purchases of equipment without maintenance; lack of interoperability; risk of excessive surveillance; applications unused by citizens.
What remained: urban sensors, mobility systems, consumption monitoring, local digital services, urban data, and useful pressure for more transparent administrations.

Blockchain
Period of maximum visibility: 2016-2021.
Initial meaning: Blockchain designates a distributed ledger maintained through cryptographic and consensus mechanisms among participants. It allows transactions to be verified without a single central authority, under certain conditions.
Dominant promise: trust without a central intermediary, auditability, and resistance to unauthorized modification.
How it became fashionable: the success of cryptocurrencies and investment enthusiasm rapidly extended the term toward logistics, health, education, electronic voting, administration, intellectual property, and digital identity.
How it became independent of mechanism: blockchain was invoked in projects where there was a clear central authority, actors did not need distributed consensus, or a traditional database was the simpler solution.
Indicator of semantic emptying: "blockchain-based" without explaining the trust problem, without reason for decentralization, without analysis of consensus cost, and without a governance model.
Possible effects of the fashion: pilot projects without users, resource consumption, complicated architectures, legal confusion, integration costs, and disproportionate promises. Blockchain sometimes became an answer before the question was formulated.
What remained: cryptocurrencies, smart contracts, distributed ledgers in limited cases, auditability, tokenization, and a more mature discussion of trust, governance, and distributed infrastructure.

Smart contracts
Period of maximum visibility: 2016-2022.
Initial meaning: smart contracts are programs that run on blockchain infrastructure and execute transactional rules deterministically when established conditions are met.
Dominant promise: automating agreements without an intermediary and reducing verification or execution costs.
How it became fashionable: with programmable blockchain platforms, smart contracts were presented as a mechanism for decentralized finance, digital property, logistics, copyright, and many other domains.
How it became independent of mechanism: the term "contract" was taken too literally. In many cases, executable code did not solve the

legal problem, the interpretation of clauses, or disputes in the real world.
Indicator of semantic emptying: "smart contract" without specifying the relationship between code, legal obligation, external data, arbitration mechanism, and responsibility for errors.
Possible effects of the fashion: exploited vulnerabilities, financial losses, problematic irreversibility, dependence on oracles, and confusion between technical automation and legal validity.
What remained: blockchain automation, DeFi, programmable transactional rules, code auditing, and interest in the relationship between software and contractual obligations.

FinTech
Period of maximum visibility: 2015-present, with intensification after the expansion of mobile payments, open banking, and digital financial services.
Initial meaning: FinTech designates the application of digital technology in financial services: payments, transfers, lending, savings, investment, insurance, regulation, identification, scoring, and banking infrastructures [9].
Dominant promise: faster, more accessible, cheaper, and more user-adapted financial services. FinTech promises to reduce frictions in the traditional financial system.
How it became fashionable: mobile applications, digital wallets, instant payments, financial startups, and open banking made the term highly visible. For many companies, the FinTech label became a sign of agility and innovation.
How it became independent of mechanism: the term came to be used for almost any application with a financial component, even when the technology was modest or the service merely reproduced a classical financial process digitally.
Indicator of semantic emptying: "FinTech solution" without a clear financial model, regulatory analysis, consumer protection, security, interoperability, and real difference from existing digital banking services.
Possible effects of the fashion: financial applications launched before risk controls mature, confusion between user experience and financial soundness, overvaluation of startups, aggressive lending products, and dependence on external payment infrastructures.
What remained: mobile payments, digital banking, open banking, regtech, insurtech, robo-advisory, alternative scoring, automation of financial processes, and real competition between traditional financial institutions and digital providers.

IoT / Internet of Things
Period of maximum visibility: 2017-present.
Initial meaning: IoT designates the connection of physical objects to digital networks for data collection, monitoring, control, or automation.
Dominant promise: the physical world becomes measurable, connected, and digitally controllable.
How it became fashionable: the low cost of sensors, wireless communications, and cloud platforms enabled many monitoring projects. The term became attractive in industry, agriculture, health, smart homes, and administration.
How it became independent of mechanism: any connected sensor was called "IoT," regardless of security, scaling, maintenance, data integration, or operational utility.
Indicator of semantic emptying: an "IoT solution" composed of sensors that send data to a dashboard, without alerting, control, integration, calibration, or a maintenance plan.
Possible effects of the fashion: insecure devices, collected but unused data, maintenance costs, batteries, unstable connectivity, and risk of abandonment after a pilot.
What remained: telemetry, remote monitoring, industrial sensors, smart homes, energy management, precision agriculture, and integration of physical data into digital applications.

Edge computing

Period of maximum visibility: 2018-present.
Initial meaning: edge computing designates data processing close to its source, at the edge of the network, to reduce latency, save bandwidth, or ensure local autonomy.
Dominant promise: fast response, local processing, and reduced dependence on a central cloud.
How it became fashionable: IoT, autonomous vehicles, industry, video analytics, and real-time applications created interest in local processing.
How it became independent of mechanism: the term was used for any local device with minimal computing capacity, even where there was no real problem of latency or traffic.
Indicator of semantic emptying: "edge AI" used for a device that runs a simple local rule or forwards data without relevant processing.
Possible effects of the fashion: local infrastructure is difficult to administer, updates are difficult to update, there is uneven security, and complexity is distributed across many poorly controlled nodes.
What remained: cloud-edge architectures, local industrial processing, inference on devices, low-latency monitoring, and autonomy in environments with unstable connectivity.

Digital twin
Period of maximum visibility: 2019-present.
Initial meaning: "digital twin" designates a digital representation connected to a physical system, updated through data, and used for monitoring, simulation, prediction, or optimization.
Dominant promise: understanding and optimizing the physical system through its digital model.
How it became fashionable: the term was adopted in industry, energy, construction, urbanism, health, and applied research. It attractively combines data, modeling, simulation, and visualization.
How it became independent of mechanism: a digital twin was used for 3D models, maps, dashboards, or static simulations without synchronization with the real system.
Indicator of semantic emptying: "digital twin" without data flow from the physical system, without a model, without updating, without simulation, and without operational decisions derived from the digital representation.
Possible effects of the fashion: expensive visual projects with limited operational value; models that look good in presentations but do not change maintenance, control, or prediction; confusion between representation and function.
What remained: advanced monitoring, simulation connected to data, predictive maintenance, cyber-physical integration, and serious applications in industrial, energy, and urban systems.

Low-code/no-code
Period of maximum visibility: 2020-present.
Initial meaning: low-code/no-code platforms allow applications, automations, or workflows to be built through visual interfaces, predefined components, and configurations, with little or no manually written code.
Dominant promise: rapid development accessible to non-programmer users.
How it became fashionable: the shortage of developers, the need for departmental applications, and the success of automation platforms made the term highly attractive to organizations.
How it became independent of mechanism: "no code" was confused with "no complexity." In reality, applications have rules, data, permissions, exceptions, and integration.
Indicator of semantic emptying: "Anyone can create applications" without discussion of security, governance, life cycle, testing, and maintenance.
Possible effects of the fashion: departmental applications hard to control, shadow IT, platform dependence, unaudited workflows, and accumulation of fragile solutions. Some organizations solve small problems quickly

but create technical debt outside the IT department.
What remained: simple automations, rapid prototyping, internal applications, integration of forms and workflows, and an old lesson: users can build useful tools, but they need rules.

RPA / Robotic Process Automation
Period of maximum visibility: 2018-present.
Initial meaning: RPA designates the use of software robots to automate repetitive, rule-based tasks, usually by imitating user actions in existing interfaces.
Dominant promise: rapid automation without rewriting legacy systems. The organization could reduce repetitive work without deep integration among applications.
How it became fashionable: many institutions and companies had repetitive processes, old systems, and limited budgets for complete modernization. RPA seemed a pragmatic solution: the robot works over interfaces without modifying core applications.
How it became independent of mechanism: the term was used for any office automation, script, or automatically triggered workflow.
Indicator of semantic emptying: "RPA" without a stable process, clear rules, treatment of exceptions, robot monitoring, and a maintenance plan for when application interfaces change.
Possible effects of the fashion: automation of poor processes, robots fragile to interface changes, accumulation of automations difficult to inventory, and the risk of postponing real system integration.
What remained: automation of repetitive tasks, pragmatic connection of legacy systems, reduction of manual copying work, and an entry point into process analysis.

Hyperautomation
Period of maximum visibility: 2020-present.
Initial meaning: hyperautomation designates the combination of several automation technologies, such as RPA, process mining, AI, machine learning, OCR, BPM, and application integration, to automate broader processes.
Dominant promise: the transition from automating isolated tasks to coordinated automation of processes.
How it became fashionable: After the success and limits of RPA, hyperautomation promised a higher stage: not just robots repeating steps, but identification, orchestration, and optimization of automatable processes.
How it became independent of mechanism: the term began to be used for collections of automation tools without process architecture and without real coordination.
Indicator of semantic emptying: "hyperautomation strategy" without a process map, prioritization, metrics, exception analysis, and relationship among people, robots, and systems.
Possible effects of the fashion: over-automation, operational fragility, licensing costs, processes automated without prior simplification, and unclear responsibilities.
What remained: the combination of RPA with process mining, AI, and BPM; more coherent automation of repetitive workflows; greater attention to process discovery before automation.

MLOps
Period of maximum visibility: 2020-present.
Initial meaning: MLOps designates practices for operationalizing machine learning models: versioning data and models, reproducibility, deployment, monitoring, degradation detection, governance, and retraining.
Dominant promise: ML models can be administered as production software, with operational discipline.
How it became fashionable: many machine learning projects remained experimental. MLOps appeared as a response to the distance between the notebook and the production system.
How it became independent of mechanism: MLOps was sometimes reduced to tracking tools, pipelines, or dashboards, without changing the delivery process.

Indicator of semantic emptying: "We have MLOps" without monitoring performance in production, versioning data, retraining criteria, and responsibility for model errors.
Possible effects of the fashion: expensive platforms, formal pipelines, models unverified in production, and the impression that a tool solves governance.
What remained: model monitoring, reproducibility, controlled deployment, continuous evaluation, and the discipline necessary to use ML in real applications.

Zero Trust
Period of maximum visibility: 2020-present, with older conceptual roots.
Initial meaning: Zero Trust designates a security approach in which access is not granted on the basis of implicit trust in a network or location, but is continuously verified based on identity, context, device, policies, and risk [5].
Dominant promise: better security in an environment with cloud, remote work, diverse devices, and increasingly weak network perimeters.
How it became fashionable: cyberattacks, cloud migration, and hybrid work reduced the usefulness of the classical model "trusted inside, unsafe outside." Zero Trust became central in security strategies.
How it became independent of mechanism: the term was applied to isolated products, authentication solutions, or general policies, without a complete architecture.
Indicator of semantic emptying: "Zero Trust" without asset inventory, managed identities, access policies, segmentation, continuous monitoring, and least privilege.
Possible effects of the fashion: products presented as Zero Trust but not integrated; policies hard to administer; user friction; the false impression that a product solves an architecture.
What remained: continuous verification, conditional access, segmentation, central identity, least privilege, monitoring, and a useful critique of implicit trust in the network.

Observability
Period of maximum visibility: 2020-present.
Initial meaning: observability designates the ability of a system to allow understanding of its internal state based on external outputs: logs, metrics, traces, events, and correlated signals [16].
Dominant promise: distributed systems that are easier to diagnose, operate, and repair.
How it became fashionable: microservices, Kubernetes, cloud, and distributed architectures made debugging more difficult. Observability appeared as a response to the insufficiency of traditional monitoring.
How it became independent of mechanism: the term began to be used for any log collection, any dashboard, or any alerting tool.
Indicator of semantic emptying: "observability" without distributed tracing, correlation of signals, clear operational questions, and the ability to diagnose previously unknown problems.
Possible effects of the fashion: high log-storage costs, multiple tools without integration, excessive alerts, unused dashboards, and the impression that more operational data automatically means understanding.
What remained: structured logs, metrics, tracing, correlation, SLOs, incident analysis, and more mature operation of distributed systems.

FinOps
Period of maximum visibility: 2021-present.
Initial meaning: FinOps designates an operational and cultural framework through which engineering, finance, and business teams collaborate to manage cloud costs and the value obtained from technology [17].
Dominant promise: the cloud can be financially controlled without blocking technical agility.
How it became fashionable: after rapid cloud adoption, many organizations discovered that bills are variable, difficult to forecast, and directly connected to everyday technical decisions.

How it became independent of mechanism: FinOps sometimes came to mean simple cost reduction or checking the cloud bill without changing the way teams design, tag, monitor, and optimize resources.
Indicator of semantic emptying: "FinOps program" without tagging, team budgets, cost allocation, forecasting, technical responsibility, and a relationship between cost and value.
Possible effects of the fashion: mechanical cost cuts that affect performance, financial reports unused by technical teams, conflicts between finance and engineering, and local optimization without strategy.
What remained: visibility of cloud costs, distributed financial responsibility, continuous optimization, governance, and a more mature discussion of the real cost of cloud architectures.

Data lake
Period of maximum visibility: 2015-present.
Initial meaning: data lake designates a large-scale data repository in which data can be stored in raw or semi-processed form, structured, semi-structured, or unstructured, for later analytical uses.
Dominant promise: the organization can store diverse data without imposing a rigid model from the beginning, applying schema and interpretation at read time.
How it became fashionable: big data, Hadoop, and then cloud platforms made the idea of a flexible repository attractive, less rigid than the data warehouse.
How it became independent of mechanism: data lake was used for any large storage space into which files are thrown without cataloging and governance.
Indicator of semantic emptying: "data lake" without a data catalog, metadata, access policies, quality, and data life cycle.
Possible effects of the fashion: data swamp, storage costs, redundant data, discovery difficulties, uncontrolled access risks, and analytical projects slowed by lack of context.
What remained: flexible storage, separation of storage from processing, preservation of raw data, integration of unstructured data, and complementarity with the data warehouse.

Lakehouse
Period of maximum visibility: 2020-present.
Initial meaning: lakehouse designates an architecture that attempts to combine the flexibility of the data lake with some properties of the data warehouse: transactions, schema, governance, performance, and support for diverse analytics [18].
Dominant promise: a unified platform for raw data, clean data, reporting, machine learning, and analysis.
How it became fashionable: organizations encountered the limits of uncontrolled data lakes and the costs of separate systems. Lakehouse promised to reduce fragmentation between data lake and data warehouse.
How it became independent of mechanism: the term was applied to cloud data repositories or hybrid architectures that did not have transactional properties, clear governance, or sufficient analytical performance.
Indicator of semantic emptying: "lakehouse" without a controlled table format, catalog, version control, quality policies, and demonstration of performance for the intended workloads.
Possible effects of the fashion: commercial repositioning of existing platforms, architectural confusion, unrealistic expectations regarding complete unification, and underestimation of governance work.
What remained: open table formats on object storage, integration of analytics with ML, better governance over the data lake, and serious discussion of less fragmented data architectures.

Data mesh
Period of maximum visibility: 2021-present.
Initial meaning: A data mesh designates a decentralized approach to analytical data management, based on domain ownership, data as a product, a self-service platform, and federated governance [19], [20].

Dominant promise: large organizations can avoid the bottlenecks of centralized data platforms by moving responsibility closer to the domains that produce and understand the data.
How it became fashionable: many organizations were dissatisfied with overloaded central data teams, disorderly data lakes, and slow delivery of analytical products.
How it became independent of mechanism: the term was used for any decentralization of data, any team reorganization, or any distributed governance initiative.
Indicator of semantic emptying: "data mesh" without responsible domain teams, data as product, self-service platform, and federated governance.
Possible effects of the fashion: fragmentation of data under the pretext of autonomy, divergent standards, unclear responsibilities, high organizational costs, and coordination difficulties across domains.
What remained: the idea of a data product, domain responsibility for data, federated governance, internal data platforms, and a useful critique of excessive centralization.

Platform engineering
Period of maximum visibility: 2022-present.
Initial meaning: platform engineering designates building and operating internal platforms that offer developers standardized services: deployment, observability, security, configuration, infrastructure, templates, and pipelines.
Dominant promise: a better developer experience and reduced operational complexity through well-designed internal platforms.
How it became fashionable: after the expansion of DevOps, cloud-native, and Kubernetes, developers were exposed to increasing operational complexity. Platform engineering appeared as a reaction: not every team needs to reinvent infrastructure.
How it became independent of mechanism: the term was used for any renamed infrastructure team, internal portal, or collection of scripts.
Indicator of semantic emptying: "platform engineering" without a clear internal product, developer users, service catalog, internal SLAs, and measurement of developer experience.
Possible effects of the fashion: internal platforms imposed rather than adopted; bureaucratization of DevOps; central teams recreating the old ticketing model; tools too rigid for team needs.
What remained: internal platforms, golden paths, deployment templates, self-service services, reduction of repetitive tasks for developers, and a more mature form of infrastructure organization.

Metaverse
Period of maximum visibility: 2021-2023.
Initial meaning: metaverse designated persistent, immersive, possibly interoperable virtual spaces in which users could work, socialize, buy, create, and interact through avatars.
Dominant promise: a new form of digital presence, with broad social and economic implications.
How it became fashionable: large technology companies invested heavily in the term, while the press and consulting amplified the idea of a new stage of the Internet.
How it became independent of mechanism: any 3D space, VR application, social game, or virtual conference room could be presented as metaverse.
Indicator of semantic emptying: "metaverse" without persistence, interoperability, community, internal economy, or utility superior to an ordinary videoconference.
Possible effects of the fashion: large investments in products without adoption, artificial virtual events, empty spaces, and confusion between a technological demonstration and user need. After 2023, industry interest visibly shifted toward generative AI.
What remained: VR/AR, simulation, immersive collaboration in niches, technical training, social gaming, and some specialized industrial or educational applications.

Web3
Period of maximum visibility: 2021-2023.
Initial meaning: Web3 designated a vision of the Web based on decentralization, blockchain, tokens, digital wallets, and user-controlled identity.
Dominant promise: an Internet less dependent on central platforms and more favorable to direct digital ownership.
How it became fashionable: cryptocurrencies, NFTs, and investments in blockchain startups turned Web3 into a term with strong ideological and financial charge.
How it became independent of mechanism: many Web3 projects were, in practice, ordinary web applications with an attached token or a marginal blockchain component.
Indicator of semantic emptying: "Web3" without real decentralization, clear governance, token utility, and explanation of the advantage over a conventional web application.
Possible effects of the fashion: speculation, abandoned projects, confusion between community and investment, transaction costs, and usability barriers for ordinary users.
What remained: digital wallets, some decentralized applications, tokenization, discussions of digital property and identity, and a healthier skepticism toward promises of total decentralization.

Privacy-enhancing technologies / PETs
Period of maximum visibility: 2021-present, with older technical roots.
Initial meaning: privacy-enhancing technologies designate techniques and tools that allow the use, sharing, or analysis of sensitive data while reducing exposure of personal or confidential information [6], [7], [8].
Dominant promise: using data without compromising privacy. Organizations could collaborate, analyze, or publish results with lower risk for the persons concerned.
How it became fashionable: data protection regulations, the growth of AI projects, and interest in interorganizational data made PETs highly attractive.
How it became independent of mechanism: the term was used generically for any privacy measure, from vague anonymization to administrative policies, without specifying the technique.
Indicator of semantic emptying: "We use PETs" without specifying whether it is differential privacy, homomorphic encryption, secure multiparty computation, synthetic data, secure enclaves, or another mechanism.
Possible effects of the fashion: a false sense of safety, reversible anonymizations, insufficiently evaluated synthetic data, high computational costs, and confusion between legal compliance and real technical protection.
What remained: differential privacy, secure computation, homomorphic encryption, synthetic data, protected execution environments, and a more mature discipline of using sensitive data.

Generative AI
Period of maximum visibility: 2022-present.
Initial meaning: Generative AI designates models capable of producing text, code, images, sound, video, or other artifacts, based on training data and inputs supplied by users.
Dominant promise: partial automation of cognitive and creative activities: writing, programming, synthesis, documentation, design, conversation, translation, and visual generation.
How it became fashionable: conversational applications and image generators made AI visible to the broad public. For the first time, many users interacted directly with advanced models without technical mediation.
How it became independent of mechanism: the term was attached to products with minimal AI functions: automatic completions, improved search, templates, or rules. Sometimes any product with a side chatbot was presented as generative AI.
Indicator of semantic emptying: "powered by generative AI" without specifying the model, accessed data, limits, evaluation, responsibility for errors, and mode of integration into the workflow.

Possible effects of the fashion: rapid purchases without governance, exposure of internal data, unverified results, premature replacement of human expertise, underestimated inference costs, and products that use AI as commercial ornament.
What remained: writing assistants, code generation, document synthesis, conversational interfaces, visual-creation tools, partial automation, and a new discipline for evaluating systems based on generative models.

Prompt engineering
Period of maximum visibility: 2022-present.
Initial meaning: prompt engineering designates formulating inputs for generative models so that the result is more controlled, more complete, or better suited to the purpose.
Dominant promise: programming through natural language. The user can obtain better results through well-formulated instructions.
How it became fashionable: with the expansion of LLMs, courses, guides, and professional roles dedicated to prompts appeared. The term was sometimes presented as a new central competence.
How it became independent of mechanism: prompt engineering was artificially separated from domain knowledge, result evaluation, understanding model limits, and workflow design.
Indicator of semantic emptying: the promise that "the right prompt" solves complex problems without data, verification, or expertise.
Possible effects of the fashion: overestimation of magic formulas, collections of prompts copied mechanically, apparently good but unverified results, dependence on unstable model behaviors.
What remained: structuring requirements, examples in prompts, defining role and format, iterative evaluation, testing on cases, and integrating prompts into broader systems.

RAG / Retrieval-Augmented Generation
Period of maximum visibility: 2023-present.
Initial meaning: RAG designates response generation by a language model augmented through retrieval of relevant documents or fragments from an external source. The aim is for the answer to be grounded in available data, not only in the model's parameters.
Dominant promise: more correct, more current responses adapted to the organization's internal knowledge.
How it became fashionable: organizations wanted to use LLMs on their own documents, procedures, policies, knowledge bases, and internal archives. RAG became the default architecture for many such projects.
How it became independent of mechanism: RAG came to designate any chatbot that searches documents, even if retrieval is weak, sources are not cited, fragments are irrelevant, or no evaluation exists.
Indicator of semantic emptying: "enterprise RAG" without controlled indexing, retrieval evaluation, treatment of outdated documents, citations, and quality criteria for responses.
Possible effects of the fashion: internal systems that answer persuasively but incorrectly; retrieval of old documents; exposure of sensitive information; maintenance costs for the index; excessive trust from users.
What remained: semantic search, grounding, source citation, internal document assistants, and a useful architecture when designed and evaluated correctly.

Copilot
Period of maximum visibility: 2023-present.
Initial meaning: copilot designates an AI assistant integrated into a work tool, offering suggestions, completions, summaries, or contextual actions.
Dominant promise: higher productivity through permanent assistance in professional applications.
How it became fashionable: the term spread from programming to office suites, CRM, BI, education, design, security, and administration.
How it became independent of mechanism: any suggestion, summarization, or chatbot

function embedded in an application began to be presented as a copilot.
Indicator of semantic emptying: a "copilot" that does not understand the application's context, cannot act safely, does not cite sources, and does not explain the limits of the result.
Possible effects of the fashion: increased licensing costs, excessive trust in suggestions, reduced user attention, and superficial integration into real workflows.
What remained: code completion, summarization, text generation, natural language querying, contextual assistance, and a new kind of interface between user and application.

Quantum computing
Period of maximum visibility: 2019-present, with older scientific roots.
Initial meaning: quantum computing designates computation based on principles of quantum mechanics such as superposition, interference, and entanglement, using qubits instead of classical bits [21].
Dominant promise: solving classes of problems that are very difficult for classical computers, especially in quantum simulation, optimization, cryptanalysis, and certain mathematical problems.
How it became fashionable: experimental progress, investments by large companies, and implications for cryptography brought the term into broad technological discourse.
How it became independent of mechanism: quantum computing was sometimes attached to almost any promise of acceleration, including AI, general optimization, or security, without specifying the quantum algorithm, hardware model, or advantage over classical methods.
Indicator of semantic emptying: "quantum-powered" without real qubits, quantum algorithm, noise analysis, error correction, and comparison with a classical algorithm.
Possible effects of the fashion: premature investments, confusion between research and commercial product, exaggerated promises regarding immediate utility, and use of the term to increase the attractiveness of projects without a real quantum component.
What remained: solid research in quantum information, quantum simulation, post-quantum cryptography, quantum algorithms, experimental hardware, and institutional preparation for future risks to cryptography.

AI-native
Period of maximum visibility: 2024-present.
Initial meaning: AI-native should designate applications designed from the start around AI capabilities, not traditional applications to which a chatbot was later added.
Dominant promise: a new generation of software in which interaction, automation, and adaptation are organized around AI models.
How it became fashionable: startups and software vendors used the term for positioning. It signals that the product is not merely compatible with AI, but built for AI.
How it became independent of mechanism: the term was also applied to ordinary products that added punctual generative functions.
Indicator of semantic emptying: "AI-native" when the application has the same architecture and workflow as before, and AI appears only in a side panel.
Possible effects of the fashion: superficial rebranding, premature purchases, unrealistic expectations, and products difficult to evaluate because the promise is larger than the available mechanism.
What remained: better-integrated conversational interfaces, contextual automations, personalization, content generation, and the beginning of a real redesign of some applications.

AI agents / agentic AI
Period of maximum visibility: 2024-present.
Initial meaning: AI agents designate systems that can plan, use tools, access data, execute actions, and adjust steps according to results, with different levels of autonomy. The idea of the agent is not new in artificial intelligence, but LLMs have made it more visible in commercial applications.

Dominant promise: automation of complex workflows, not merely generation of responses. The agent does not only answer questions; it acts.
How it became fashionable: after the wave of conversational assistants, the term "agent" promised a transition from conversation to execution. Vendors began to present agents for sales, support, programming, analysis, HR, finance, and operations.
How it became independent of mechanism: chatbots with access to an API, automated scripts, or fixed workflows were called agents even when they did not plan, monitor results, or have real autonomy.
Indicator of semantic emptying: "agentic AI" without a specified level of autonomy, controlled permissions, logging, action limits, evaluation, and recovery mechanisms in case of error.
Possible effects of the fashion: operational risks, excessive access to systems, wrong actions executed quickly, high costs, unclear value, and abandoned projects. Gartner predicted that many agentic AI projects could be canceled by the end of 2027 because of cost, unclear value, or insufficient risk controls [22].
What remained: tool orchestration, automation of repetitive tasks, supervised agents, workflows with human approval, classification of levels of autonomy, and the need for precise controls of access, audit, and responsibility.

6 Effects of terminological fashion on IT projects

Terminological fashions do not remain at the level of language. They change priorities, budgets, procurements, careers, research topics, and software architectures. A prestigious term can concentrate attention on a real problem, but it can also distort the way the problem is understood. In computer science, the effect is more severe than in many discursive fields, because words are quickly transformed into purchased infrastructures, designed systems, and reorganized processes.

Excessive focus

The first consequence is excessive focus. An organization comes to view all problems through the dominant term of the period. If the term is "cloud," the problem seems to be the location of infrastructure. If the term is big data, the problem seems to be data volume. If the term is AI, the problem seems to be the absence of an automatic model. If the term is "Zero Trust," the problem seems to be access alone. If the term is "data mesh," the problem seems to be the decentralized organization of data. If the term is FinTech, the problem seems to be the replacement of the traditional financial intermediary by a more agile application.
Focus can be useful when a field is scattered. A common term allows actors to be mobilized and reduces initial confusion. Yet it becomes dangerous when other diagnoses are eliminated prematurely. An organization with incoherent data does not first need big data, data lake, or lakehouse, but rules for data quality. An institution with unclear procedures does not primarily need digital transformation but a description of processes. A slow application does not automatically need the cloud, but profiling, correct architecture, optimized queries, and understanding of the real workload do. An insecure system does not become secure by invoking Zero Trust, but through an inventory of assets, access policies, segmentation, monitoring, and identity administration.
This focus produces selective blindness. When a term dominates, uncomfortable questions appear secondary: who will maintain the system, what data are missing, how the benefit is measured, what processes change, what competencies exist, what costs appear after the first year, and what new vulnerabilities are introduced. The project becomes coherent at the level of vocabulary but fragile at the level of execution.

Disproportionate spending

A second consequence is disproportionate spending. Fashionable terms justify larger budgets because they are associated with

modernization, competitiveness, or alignment with the dominant directions of the market. This association can be legitimate. There are situations in which the cloud reduces delivery time, the data warehouse clarifies reporting, and AI increases efficiency in a task. The problem appears when the budget is determined by the prestige of the term, not by an analysis of need.

Disproportionate spending takes several forms. It can mean buying a platform before defining use cases. It can mean expensive licenses for functions used only marginally. It can mean oversized infrastructure for modest data volumes. It can mean extensive consulting for a problem that could have been solved by simplifying a process. It can also mean premature migration to an architecture more complex than the real need.

In data projects, this tendency appears in the acquisition of data lakes without governance, lakehouses without use cases, or data mesh platforms without domains capable of assuming data ownership. In the cloud, it appears in the form of recurring costs that are hard to anticipate. FinOps became prominent precisely as a reaction to this problem: cloud promised elasticity and pay-per-use but also introduced distributed, variable, hard-to-control costs [17]. In AI, disproportionate spending appears through licenses, integration, inference consumption, security, evaluation, and human verification. The visible cost is often only the beginning. The real cost includes operation, updating, quality control, staff training, and the repair of wrong decisions.

Stillborn projects

Some projects are stillborn. They do not fail because implementation is weak, but because the initial idea has no real need behind it. They were formulated in order to use a term, not to solve a problem. These projects can have good documentation, diagrams, budgets, vendors, and even prototypes. What they lack is practical tension: a user who needs the result, a decision that must be made, a process that must be shortened, an error that must be reduced, a cost that must be controlled.

A blockchain project in an organization with a clear central authority can be stillborn if decentralization adds nothing. A digital twin without operational data, without a model, and without a technical decision is stillborn even if it looks good in a demonstration. A RAG chatbot on internal documents is stillborn if the documents are old, contradictory, and no one is responsible for updating them. An AI project for prediction is stillborn if the organization has no clean history and cannot change the process based on the prediction. A quantum computing project is stillborn if it cannot indicate the problem, the algorithm, and the plausible quantum advantage.

Stillborn projects consume resources and produce a more serious effect: institutional cynicism. After several failed initiatives, staff begin to treat every new technology as a passing fashion. When a genuinely useful solution appears, it is met with the skepticism produced by earlier failures.

Architectures adopted through imitation

Computer science has a strong culture of success models. Practices used by large companies are taken as benchmarks: microservices, Kubernetes, data lakes, streaming, observability, platform engineering, MLOps, feature stores, and data mesh. These practices can be excellent in the context that produced them. They become problematic when imitated without the original constraints.

A company with millions of users, autonomous teams, and services delivered daily needs a certain architecture. A small institution with a few internal applications and modest traffic can be burdened by the same architecture. Microservices can solve the rigidity of a large monolith but can turn a simple application into a distributed system that is hard to debug. Kubernetes can manage containerized applications at scale, but it can be an unnecessary operational burden for a few stable applications. A data lake can store diverse data at scale but can become a swamp

of files if cataloging and access rules are missing. A data mesh can reduce the bottlenecks of a central data team but can fragment the organization if domains lack maturity, resources, and responsibility.

Architectural imitation is fed by a mistaken reflex: if a technology is used by highly performing organizations, then it must produce performance. In reality, the technology is often the effect of a certain scale, not its cause. The order is reversed: large companies did not become large because they have microservices; they adopted microservices because certain limits of the monolith became intolerable at their scale.

Opportunistic relabeling

A terminological fashion also produces relabeling. Old activities are described with new terms in order to appear current. Reporting becomes data analytics. An automation script becomes AI. A portal becomes a digital transformation platform. A simulation becomes a digital twin. A rule-based system becomes an intelligent agent. A document collection becomes a knowledge base or a RAG-ready repository. An infrastructure team becomes platform engineering. A cloud cost audit becomes FinOps. A multi-factor authentication policy becomes Zero Trust.

Relabeling is not always fraudulent. Sometimes a new term offers a better perspective on an older practice. Yet when the label is used to obtain funding, attention, or prestige without changing the mechanism, a problem of conceptual integrity appears. In research, this phenomenon is visible in titles and abstracts. Ordinary statistical methods are redescribed as AI. Classical information systems are presented as digital transformation. Database systems become big data platforms, even when there is no volume, velocity, or variety.

Relabeling has an epistemic cost. It makes it harder to understand what was actually done. The reader must translate prestige vocabulary into verifiable mechanisms. If that translation cannot be performed, the term functions as a screen.

Failure through excessive centralization

Some fashions favor centralization. ERP, data warehouse, SOA, national platforms, government cloud, single registries, single identity, and single electronic health record: all can have serious justifications. Yet centralization becomes risky when it tries to solve through a single system organizational, legal, and operational differences that have not been harmonized.

This risk also appears in data projects. A central data warehouse can clarify reporting but can become a bottleneck if all domains wait for a single central team to clean, understand, and model their data. Data mesh appeared as a reaction to this limit but can fall into the opposite error: decentralization without common standards. The oscillation between centralization and decentralization is a sign that the problem is not only technical. Data have owners, rules, local meanings, and operational consequences.

Terms such as "integration," "single platform," or "federated governance" must be treated cautiously. Integration is not only data exchange. It presupposes compatible processes, governance, institutional acceptance, responsibilities, and adaptation of the system to local practice. Without these, the single system can become a solution that does not fit reality, while decentralization can become an excuse for incoherence.

Failure through premature automation

Other fashions favor automation. RAD, workflow, BPM, RPA, hyperautomation, AI, agents, smart contracts, and automated decision systems promise to reduce human intervention. This promise is seductive because people are costly, slow, inconsistent, and hard to coordinate. Yet automating a poorly understood process produces faster errors, not better results.

RPA is the clearest example. If the process is stable, repetitive, and rule-based, a software robot can reduce manual work. If the process is ambiguous, has many exceptions, or

depends on interpretation, the robot becomes fragile. Hyperautomation amplifies the same tension. Instead of automating a single task, it promises the automation of whole chains of activities through RPA, AI, process mining, and orchestration. Without process analysis, the term merely expands the error surface.

The lesson is simple: automation requires the stability of the process being automated. If the flow, rules, exceptions, and responsibilities are unclear, the technical system becomes the place where all ambiguities meet. In current language, an AI agent executing actions in a poorly defined process will amplify disorder, not eliminate it.

Failure through uncontrolled operational speed

Some technologies increase system speed. Automated trading, CI/CD, autoscaling, RPA, AI agents, and serverless automations can execute many actions in a short time. Speed is valuable, but it reduces the time available for human intervention. If an error enters such a system, it can propagate before anyone understands what is happening.

This lesson is relevant to the current wave of AI agents. A chatbot that gives a wrong answer produces a limited risk if the user verifies the result. An agent that sends emails, modifies databases, launches commands, makes purchases, or changes configurations produce a different kind of risk. Autonomy must be accompanied by limits, logs, approvals, simulation, testing, and the possibility of rapid shutdown.

Observability is important precisely here. Distributed systems, automations, and agents must be understandable through the signals they emit. Yet observability becomes an empty fashion if reduced to massive log collection. More operational data do not automatically mean better diagnosis. Operational questions, correlation, tracing, relevant metrics, and response procedures are required [16].

Failure through excessive cognitive promise

AI has a particular feature. It does not promise only technical efficiency but the partial substitution of human judgment. This promise is more sensitive than the promise of a database or cloud infrastructure. When an AI system is presented as an expert, doctor, adviser, teacher, programmer, or autonomous agent, expectations rise rapidly.

The lesson is not that AI has no place in high-risk domains. On the contrary, there are areas where AI methods produce important results: imaging, triage, information extraction, decision support, assisted programming, and document synthesis. The lesson is that the term AI cannot replace validation, integration with workflow, professional responsibility, and error analysis. In high-risk domains, cognitive promise must be verified more strictly than operational promise. The difficulties surrounding highly publicized medical AI projects illustrate the gap between promise and validated practice [23].

The same caution applies to related terms: AI-native, copilot, RAG, and agentic AI. A copilot that suggests can be useful. An agent that acts must be controlled. A RAG system that cites sources can help. A RAG system that retrieves outdated documents can produce wrong answers with an appearance of authority. An AI-native product can be a real redesign of the application. It can also be a chatbot added to a corner of the interface.

Failure through confusion between sustainability and image

Green IT and, more recently, discussions about the energy consumption of AI introduced another form of terminological fashion: technological sustainability. The term is necessary. Data centers, the training of large models, device consumption, and electronic waste have real effects. Yet the vocabulary of sustainability can be used decoratively.

A Green IT strategy is weak if reduced to reducing paper, buying "efficient" equipment, or presenting cloud migration automatically as ecological. Consumption must be measured. The life cycle of equipment must be analyzed. Server

utilization, cooling, energy, recycling, and duration of use must be treated as concrete variables. Otherwise, the term "green" becomes a moral ornament.
This problem will become more visible in the context of generative AI. Large models have training and inference costs. If an organization introduces AI for minor tasks without evaluating the benefit, it may consume resources for marginal gain. Sustainability does not require rejecting technology but proportionality.

Failure through overhyped scientific promise
Quantum computing is a special case. Unlike many commercial labels, its scientific basis is profound. Quantum computing is a real field, with algorithms, experimental hardware, open problems, and important implications for cryptography and simulation [21]. Precisely for that reason, it is vulnerable to overstatement.
The term becomes problematic when it is used as a synonym for "very fast" or "inevitable future." Not every optimization problem will be solved by a quantum computer. Not every AI project becomes better through the quantum label. Not every organization needs an immediate quantum computing strategy, although some must track cryptographic risks and the transition toward post-quantum cryptography [24].
The indicator of maturity is the ability to specify the problem, algorithm, hardware model, state of technology, and comparison with classical methods. Without these, quantum computing becomes a promise used to obtain attention, not a technical solution.

7 Typology of stillborn projects

A stillborn project is one that can be launched, funded, and even partially implemented, but lacks the minimum conditions to produce value. In computer science, such projects often appear when the fashionable term precedes the problem.

The project without a real user
This project has a formal beneficiary, but no practical user. It is written for strategy, audit, funding, or image. Real users were either not consulted or do not recognize the problem described. This category includes unused portals, rarely downloaded institutional mobile applications, dashboards no one consults, chatbots answering questions the public does not ask, and metaverse spaces without real activity.
The warning sign is the absence of a usage scenario. If one cannot describe a concrete person, with a concrete task, at a concrete moment, the project must be reconsidered.

The project without sufficient data
This type appears especially in AI, data science, big data, digital twin, RAG, data mesh, and FinTech. Documentation describes models, predictions, simulations, recommendations, and scoring, but the data are missing, incomplete, dirty, or legally inaccessible.
The warning sign is the phrase "the data will be collected later," when the entire value of the project depends on data. In some projects, data collection is a legitimate stage. In others, it is a risky assumption hidden under analytical vocabulary.

The project without a changed process
This project produces an instrument, but does not change the way work is done. A reporting system does not matter if decisions are made the same way as before. A predictive model does not matter if no one changes action based on prediction. A portal does not matter if the citizen still has to go to the office. An RPA automation does not matter if it merely reproduces a useless process.
The warning sign is the absence of a sentence such as "after implementation, process X will unfold as follows." If the new tool does not modify the real flow, it is probably decorative.

The project with disproportionate technology
In this case, the technology is more complex than the problem. Microservices are used for a small application, blockchain for a centralized database, Kubernetes for two stable services, big data for a few tens of

thousands of records, AI for simple rules, data mesh for an organization without mature data domains, and quantum computing for optimizations that have not yet been attempted with classical methods.
The warning sign is the inability to justify the technology through measurable constraints: volume, latency, number of users, team independence, audit requirements, risk, frequency of changes, costs, and limits of simpler solutions.

The eternal pilot project
The eternal pilot is launched to demonstrate the organization's interest in a technology. There is, however, no clear path to production, operating budget, institutional owner, or integration with existing systems. After the initial presentation, the project remains in an intermediate space.
This type is frequent in blockchain, smart cities, digital twins, AI, RPA, PETs, and quantum computing. The pilot can show that something is technically possible, but not that the organization can operate, pay for, secure, and use it.
The warning sign is the lack of criteria for moving from pilot to operation. A useful pilot must have a hypothesis, duration, metrics, a decision to continue or stop, and someone responsible for the next phase.

The rebranding project
The rebranding project takes an existing activity and renames it. A report becomes analytics, an online form becomes digital transformation, a search engine becomes AI, a simulation becomes a digital twin, a script becomes RPA, an infrastructure team becomes platform engineering, and a cost audit becomes FinOps. It is not wrong for an old project to be reinterpreted through a new concept if the mechanism changes. It becomes problematic when the change is only lexical.
The warning sign is the absence of technical differences between the old version and the renamed version.

The terminological-compliance project
This project appears especially in grants, strategies, and institutional documents. The term is introduced because it is expected by the funder, evaluator, leadership, or market. The project may have a real problem, but it is rewritten to fit the vocabulary of the period.
The warning sign is the accumulation of prestigious terms that are not simultaneously necessary: cloud, AI, blockchain, digital twin, data mesh, Zero Trust, Green IT, PETs. When too many fashions appear in the same project, the text probably pursues compliance rather than precision.

8 Indicators of semantic emptying
Semantic emptying can be detected through simple questions. If the term does not withstand these questions, it functions more as an ornament than as a technical notion.

General indicators
A term is probably emptied of content when it:

- is used before the problem is described;
- cannot be tied to a concrete architecture;
- has no success criteria;
- does not specify the user;
- does not indicate the data used;
- does not explain the process changed;
- does not identify new risks;
- has no operational owner;
- does not distinguish between prototype and production;
- cannot be explained without synonyms that are equally vague.

These signs do not automatically prove that the project is weak. They do, however, indicate the need for closer examination.

Indicators specific to data
For terms such as data warehouse, data lake, lakehouse, data mesh, big data, data science, RAG, and AI analytics, semantic emptying appears when several questions cannot be answered.
What data sources are used? Who is responsible for their quality? What history exists? What data are missing? How are

contradictions handled? What decision is made based on the result? How is the usefulness of the analysis verified? Who owns the data? What is their life cycle?
Without these answers, the term "data" produces an impression of objectivity, but guarantees nothing.

Indicators specific to architecture and infrastructure
For cloud, microservices, containers, serverless, edge, Kubernetes, platform engineering, and grid computing, the questions are different.
What technical constraint justifies the architecture? What traffic volume exists? How many teams work independently? What downtime is acceptable? Who operates the system? What total cost is estimated? What is simplified and what is complicated? What competences exist internally?
If the answers are missing, the architecture is probably being adopted by imitation.

Indicators specific to cost and operation
For FinOps, observability, DevOps, MLOps, and platform engineering, the term must be tied to measurable operational practices.
How are costs allocated? Who sees them? Who can reduce them? What metrics are tracked? What incidents are analyzed? What models are monitored in production? What internal services are offered to teams? How is platform quality measured?
Without these answers, operational terms become new names for old administrative functions.

Indicators specific to security and privacy
For Zero Trust, PETs, confidential computing, and privacy by design, the term must be tied to threats, policies, and techniques.
What risk is reduced? What assets are protected? What identities exist? How is access verified? What data are sensitive? What protection technique is applied? What attack remains possible? How is the system audited?
Without these answers, security becomes a declaration, not an architecture.

Indicators specific to AI
For AI, machine learning, generative AI, RAG, copilot, AI-native, and AI agents, the term must be tied to model, data, and evaluation.
What model is used? What data enters the system? What outputs does it produce? How is quality measured? What types of error are acceptable? Who verifies the result? What actions can the system execute? What is it not allowed to do? How is it stopped or corrected? What happens when the model does not know?
In the absence of these answers, AI remains a promise. In high-risk domains, that promise is not enough.

Indicators specific to digital and sectoral transformation
For digital transformation, smart cities, Industry 4.0, Green IT, FinTech, EdTech, HealthTech, and GovTech, the central question concerns the process and the sector.
What was done before? What is done after? What time is reduced? What cost decreases? What error disappears? What step is eliminated? What service becomes accessible? What indicator changes? What regulations apply? What risk for the user appears?
If the answer consists only of a platform, application, or purchase, transformation is incomplete.

Indicators specific to emerging technologies with uncertain maturity
For quantum computing, post-quantum cryptography, AI agents, and some PETs, technological maturity must be examined explicitly.
What is the stage of the technology? Is there a product or only a prototype? Is there a demonstrated advantage over the classical method? What hardware or mathematical conditions are necessary? What limitations are known? What realistic horizon of use exists?

Without these answers, the term can function as a future promise used to justify present decisions that are not warranted.

9 Effects on research and publications

Terminological fashion also affects research. Article titles, abstracts, keywords, and grant topics align with dominant terms. This adaptation is sometimes legitimate. A new method can change a field. An emerging concept can create real research questions. Yet there is also superficial adaptation.

Repackaging old research

Existing research is renamed to fit the discourse of the moment. Statistical analyses become AI. Simulation models become digital twins. Information systems become digital transformation platforms. Applications with sensors become Industry 4.0 or smart cities. Studies on energy consumption become Green IT. Ordinary financial applications become FinTech. Optimization experiments are presented as preparation for quantum computing, although they use no quantum algorithms.

This repackaging increases visibility but can reduce scientific precision. A good article must say exactly what is new: the data, the method, the architecture, the evaluation, the application context, or the result. If the novelty is only the term, the contribution is weak.

Keywords as visibility strategy

Keywords have become instruments of visibility. Authors know that current terms increase the chances that an article will be found, cited, or sent to interested reviewers. This practice is not automatically incorrect. It becomes problematic when the term in the keywords has no real role in the content.

A study using logistic regression does not become AI merely by including the term in the title. A reporting system does not become big data if volume and variety do not require specific techniques. A chatbot does not become an agent if it has no autonomy, planning, and controlled tool use. A study on cloud costs does not become FinOps if it does not address the operational relationship among engineering, finance, and business.

Evaluators and funders captive to vocabulary

Funders and evaluators are exposed to the same dominant vocabulary. Sometimes a project seems more competitive if it uses the expected terms. This creates an incentive to include them even when they are not necessary. In the long term, the result is an artificial convergence of projects: many proposals use the same expressions regardless of the real differences among domains.

A healthy evaluation must ask for terms to be translated into mechanisms. If the project says "AI," one must ask what model, what data, and what evaluation. If it says blockchain, one must ask why decentralization is necessary. If it says "digital twin," one must ask what physical system is connected, what data flow there is, and what decision is made. If it says Zero Trust, one must ask what policies and verification points exist. If it says Green IT, one must ask what consumption is reduced. If it says FinTech, one must ask what the financial service is, what the risk is, and what regulation applies.

Bibliometrics and terminological waves

Many fashions can be studied bibliometrically. The frequency of terms in titles, abstracts, and keywords often indicates the appearance, peak, and decline of a wave. Semantic Web, grid computing, big data, blockchain, data mesh, lakehouse, digital twin, metaverse, generative AI, agentic AI, Zero Trust, FinOps, RPA, hyperautomation, quantum computing, and privacy-enhancing technologies can be analyzed in this way.

Such an analysis must be carried out carefully. Frequency does not directly measure scientific value. It measures the visibility of a term. However, correlated with publication types, domains, citations, and funding, it can show how vocabulary influences the research agenda.

10 What can be done: conceptual hygiene

It is neither realistic nor useful to eliminate new vocabulary. Computer science evolves through new terms because new architectures, methods, and practices appear. The aim is not terminological purism. The aim is conceptual hygiene.

Local definition of the term

A fashionable term must be defined locally. It is not enough to say that the system uses AI, cloud, Zero Trust, or RAG. One must specify what the term means in that project.

Weak formulation:

"The platform uses AI to improve user activity."

Better formulation:

"The platform uses a language model to summarize requests received from users. The summary is displayed to the operator, who can accept, modify, or reject it. The system does not send answers automatically."

The second formulation reduces vague prestige but increases trust. It says what the system does and what it does not do.

Linking the term to mechanism

Every term must be tied to an observable mechanism. For data warehouse, the mechanism includes source integration, history, the data model, and ETL/ELT processes. For cloud, it includes provisioning, scaling, service model, and shared responsibility. For digital twin, it includes synchronization with the physical system, the model, and operational use. For Zero Trust, it includes identity, policies, continuous verification, and segmentation. For FinOps, it includes cost allocation, consumption visibility, team responsibility, and continuous optimization. For AI agents, it includes planning, tool access, action limits, monitoring, and the possibility of intervention.

If the mechanism cannot be described, the term should not be used or should be replaced with a more modest one.

Comparing with the simple solution

Every fashionable technology must be compared with a simpler solution. This comparison is often absent because the fashionable solution seems implicitly superior.

The useful questions are direct. Can the problem be solved with an ordinary database? Is blockchain necessary? Is an AI model necessary, or is a set of rules enough? Is Kubernetes necessary, or is a simple managed service enough? Is a digital twin necessary, or is a monitoring system sufficient? Is data mesh necessary, or must central data governance first be repaired? Is quantum computing necessary, or have classical methods not yet been exhausted? Is a new application necessary, or must the existing process be modified?

The simple solution is not always the correct one. It is, however, the comparison term that prevents excess.

Establishing success criteria

A project formulated through a fashionable term must have verifiable success criteria. Without them, success is confused with artifact delivery: the platform exists, the dashboard opens, the chatbot answers, and the model runs.

Criteria can be technical, operational, economic, or ecological: reduced processing time, fewer errors, higher resolution rate, lower cost per transaction, increased availability, shorter response time, improvement in a predictive metric, fewer steps for the user, reduced energy consumption, and lower risk of unauthorized access.

The fashionable term must be a means, not a result.

Separating demonstration from production

Many current technologies are spectacular in demonstrations. Generative AI, agents, visual digital twins, VR applications, interactive dashboards, PETs, and quantum prototypes can produce an immediate impression. Production is different. It requires clean data, access rights, security,

monitoring, error handling, performance, predictable costs, and support.
A demonstration answers the question, "Can it be done?" Production answers the question, "Can it work repeatedly, safely, at acceptable cost, with real users?" The difference between these two questions must be preserved.

Introducing stopping mechanisms
Technologies that automate actions must have mechanisms for stopping, limiting, and auditing. This requirement is old in critical systems but becomes more important with AI agents, RPA, smart contracts, and serverless automations.
A system that can execute actions must have permission limits, logging, thresholds, human approval for risky actions, testing in an isolated environment, and rollback procedures. Without these, autonomy is a risk hidden by the vocabulary of efficiency.

Treating cost as part of architecture
Cost is not a subsequent administrative detail. In the cloud, serverless, generative AI, observability, data platforms, and MLOps, cost is a direct consequence of architecture. A technical choice can increase cost through traffic, storage, inference, logs, replication, latency, queries, or retraining.
FinOps expresses precisely this shift: cost must be visible to the teams that produce it. If engineers do not see costs and finance does not understand the architecture, optimization becomes conflictual. Serious conceptual hygiene includes total cost of ownership, not only the initial price.

Treating security and privacy as mechanisms, not labels
Zero Trust, PETs, privacy by design, and confidential computing must be tied to concrete threats. There is no security by label. There are policies, controls, identities, cryptography, segmentation, audit, limitation of privileges, and response procedures.
In projects involving personal data, privacy must be evaluated before analysis, not after the system is built. PETs can be useful, but they are not universal shields. Each technique has costs, limits, and assumptions. Weak anonymization, insufficiently verified synthetic data, or generically invoked encryption can produce false safety.

11 Recommended formulation for evaluating a fashionable term

An organization, professor, evaluator, or researcher can use a short grid before accepting a prestigious term in a project.

1. What concrete problem is formulated?
2. What fashionable term is used?
3. What is the technical meaning of the term in this project?
4. What concrete mechanism supports it?
5. What simpler solution has been compared?
6. What data are needed?
7. What process changes?
8. Who uses the result?
9. What new risks appear?
10. What initial and recurring costs appear?
11. What security, audit, and stopping mechanisms exist?
12. How is success measured?
13. What happens if the project does not produce value?
14. What remains useful if the term loses its prestige?

The last question is the most severe. If the answer is "nothing," the project is probably dependent on fashion. If cleaner data, clearer processes, reusable code, better-managed infrastructure, new competencies, better security, more visible costs, or better decisions remain, the project can be justified even after the term passes.

12 Recent terms that may follow the same trajectory

Some recent expressions are still in their ascent phase. It cannot yet be said whether they will become stable concepts, specialized technical terms, or worn-out formulas. Nevertheless, several signs are already visible: large promises, unstable definitions,

rapid inclusion in commercial presentations, presence in institutional strategies, and extension into domains where the technical mechanism is not always clear. Gartner placed AI agents and AI-ready data among the AI technologies advancing rapidly and situated at a high level of expectations in 2025, which indicates precisely this zone of terminological risk [25].

Agentic AI / AI agents 2.0
The term has technical substance when it designates systems that plan, use tools, execute steps, verify results, and operate with controlled levels of autonomy. The risk of wear appears when any chatbot connected to an API is called an agent. The indicator of semantic emptying is the absence of autonomy levels, permissions, logging, action limits, and recovery mechanisms. Recent industry discussions have already warned that one-size-fits-all agent governance can lead to failure [26].

AI-ready data
The term is useful if it means clean, documented, governed data, available under controlled access and suitable for training, inference, RAG, or automatic analysis. It can become, however, a vague formula for any data-cleaning initiative. The indicator of semantic emptying is the expression "AI-ready data" without catalog, metadata, measured quality, access rights, provenance, updating, and evaluation of adequacy for the AI task.

AI governance
The term is necessary, especially in organizations that use generative models, agents, automated decision systems, or predictive models integrated into processes. The risk is turning it into a set of declarative policies. The indicator of semantic emptying appears when AI governance is reduced to a document, without model inventory, risk classification, evaluation, monitoring, responsibilities, audit, and withdrawal procedures [27], [28].

AI TRiSM / AI trust, risk, and security management
The term is useful if it coherently brings together trust, risk, and security in AI systems. It can become, however, an overly broad umbrella used for any discussion of responsible AI. The indicator of semantic emptying is the lack of connection among risk, technical controls, evaluation, monitoring, and operational responsibility.

Responsible AI / trustworthy AI
These terms have normative value, especially in domains with impact on people. The risk is vague moralization: the system is called responsible because the project text contains ethical principles. The indicator of semantic emptying is the absence of bias tests, explanations, traceability, human evaluation, and mechanisms for contesting decisions.

Sovereign cloud
The term has gained visibility through concerns about data sovereignty, vendor dependence, and national or European regulation. It makes sense when it describes legal, operational, and technical control over data, metadata, cryptographic keys, and infrastructure location. It becomes a slogan when it means only hosting in a certain country. The indicator of semantic emptying is the absence of an answer to the question: who actually controls the data, keys, administration, and vendor access?

Confidential computing
The term designates the protection of data during processing, usually through hardware-isolated execution environments. It is relevant for interorganizational collaboration, sensitive data, and the cloud. The risk is using it as a generic synonym for advanced security. The indicator of semantic emptying is the absence of the execution environment, threat model, keys, attacks excluded, and the limits of the technique.

Post-quantum cryptography

The term is technically solid because cryptographic transition is a real problem in the perspective of quantum computing. The risk is turning it into a commercial label applied to security products. The indicator of semantic emptying is the expression "quantum-safe" without algorithms, standards, migration plans, cryptographic inventory, and analysis of compatibility with existing systems [24], [29].

Quantum AI
This is one of the terms most vulnerable to overstatement. Serious research can exist at the intersection of quantum computing and machine learning, but the term is frequently used to combine two major sources of prestige: "quantum" and "AI." The indicator of semantic emptying is the absence of a quantum algorithm, a suitable problem, a demonstrable advantage, and a comparison with classical methods.

Data fabric
The term promises data integration through active metadata, connectivity, governance, automation, and unified access. It can be useful as an architecture for organizations with many data sources. The risk is that it becomes a newer synonym for data integration or for a commercial catalog. The indicator of semantic emptying is the absence of active metadata, access policies, data provenance, and real automation.

Data products
The term is useful in data mesh and modern data architectures because it obliges domains to treat data as consumable, documented, maintained products. The risk is that any table, API, or report is renamed a "data product." The indicator of semantic emptying is the absence of an owner, data contract, quality level, documentation, and identified users.

Vector database
The term has real technical grounding in semantic search, RAG, and applications with embeddings. The risk is the purchase of a vector database where classical text search, a relational database, or an ordinary index would suffice. The indicator of semantic emptying is a project "with vector database" without retrieval evaluation, relevance metrics, and comparison with simpler alternatives [30], [31].

Knowledge graph 2.0
Knowledge graph has technical substance when there are entities, relations, ontologies, rules, querying, and maintenance. The LLM wave brought the term back into attention. The risk is that any collection of indexed documents is called a knowledge graph. The indicator of semantic emptying is the absence of the graph proper: explicit entities, semantic relations, schema, and querying.

Model Context Protocol / MCP
The term is recent and may become important as a mechanism for connecting AI models to tools and data sources. The risk is that any API integration for LLMs is renamed MCP or "MCP-ready." The indicator of semantic emptying is the absence of an implemented protocol, access control, tool descriptions, logging, and isolation among actions [32], [33], [34].

Synthetic data
Synthetic data can be useful for testing, privacy, simulation, and training under certain conditions. The risk is presenting them as a universal substitute for real data. The indicator of semantic emptying is the absence of evaluation of fidelity, reidentification risk, bias, and model performance on real data [35], [36], [37].

Digital humans / AI avatars
The term promises more natural conversational interfaces, with voice, facial expressions, and human appearance. It can be useful in training, support, or education. The risk is replacing functionality with visual effect. The indicator of semantic emptying is the avatar that impresses visually but does not solve the task better than a text chatbot.

Spatial computing
The term makes sense when it designates the interaction between digital applications and physical space through AR, VR, sensors, 3D maps, and immersive interfaces. It may repeat part of the fate of the metaverse if used for any AR application. The indicator of semantic emptying is the absence of functional advantage over 2D interfaces [38], [39].

Ambient computing
The term designates computing discreetly integrated into the environment, through sensors, connected devices, and contextual interactions. The risk is extreme vagueness: almost any smart device can be included. The indicator of semantic emptying is the inability to specify what context is perceived, what action is triggered, and what value it has for the user [40], [41].

Composable enterprise / composable architecture
The term promises organizations and systems made of flexible, recombinable components. It is related to APIs, services, modularity, and enterprise architecture. The risk is turning it into a synonym for "flexible." The indicator of semantic emptying is the absence of real components, contracts, governance, and recombination capacity [42].

API economy
The term makes sense when APIs become products, integration channels, and sources of economic value. It becomes vague when any application with an API is presented as part of the API economy. The indicator of semantic emptying is the absence of external consumers, documentation, contracts, access models, and measurement of use [43], [44], [45].

Superapp
The term designates an application that integrates many services into a single space: payments, messaging, commerce, transport, and financial services. The risk is using it for crowded applications without a real ecosystem. The indicator of semantic emptying is an application with many menus but without integrated services, partners, and recurrent use.

Embedded finance
This may be the next overhyped FinTech term. It makes sense when financial services are integrated directly into non-financial applications: payments, credit, insurance, installments, and accounts. The risk is the aggressive expansion of financial services without transparency and risk control. The indicator of semantic emptying is the absence of regulatory analysis, responsibility toward the customer, and credit or fraud risk [46].

RegTech
The term designates the use of technology for compliance, reporting, monitoring, and control of regulatory risk. It is useful in finance and regulated domains. It can, however, become a label for any reporting application. The indicator of semantic emptying is the absence of explicit correspondence with regulatory requirements, controls, audits, and traceability.

Open banking / open finance
The terms have regulatory and technical grounding where access to financial data is permitted through APIs and consent. The risk is using them for any financial integration. The indicators of semantic emptying are the absence of controlled consent, API security, interoperability, and a legal framework [47], [48], [49].

Autonomous enterprise
The term is highly exposed to inflation. It suggests organizations in which processes execute, adjust, and optimize themselves with reduced human intervention. It may designate a serious direction of automation, but the risk is evident: any combination of AI agents, RPA, and dashboards can be presented as organizational autonomy. The indicator of semantic emptying is the absence of concrete autonomous processes, decision

limits, and mechanisms of responsibility [50].

Agentic process automation
The term combines RPA, hyperautomation, and AI agents. It can designate automation of processes in which agents plan and use tools, but it can quickly become a relabeling of older process automation. The indicator of semantic emptying is the absence of difference from classical RPA: if the process is fixed, rule-based, and without planning, the term "agentic" is probably decorative.

Decision intelligence
The term promises to connect data, models, rules, and decision processes. It is useful if it obliges the mapping of decisions and the evaluation of effects. The risk is that it becomes a new label for BI or analytics. The indicator of semantic emptying is the absence of the concrete decision that changes.

Autonomous cybersecurity / preemptive cybersecurity
The terms have become more visible against the background of AI and proactive security. Gartner included preemptive cybersecurity among technology trends for 2026, with the use of AI to anticipate and block threats before they materialize [51]. The risk is presenting traditional detection and response mechanisms as complete autonomy. The indicator of semantic emptying is the absence of real autonomy, action limits, reduced response time, and evaluation of false positives.

Green AI / sustainable AI
The term may become important because large models consume computing resources, energy, and water for cooling. It makes sense when it includes measuring consumption, inference efficiency, choosing the appropriate model, and reporting impact. It becomes a slogan when "sustainable AI" only means using AI for environmental topics. The indicator of semantic emptying is the absence of data about consumption and the absence of comparison among models.

Small language models / domain-specific LLMs
The term has stable potential as a reaction to the costs and risks of large models. The risk is that it becomes a label for poorly evaluated models promoted as cheap and private. The indicator of semantic emptying is the absence of benchmarks on real tasks, comparison with large models, and total-cost analysis.

AI PC / on-device AI
The term indicates the local running of models on personal devices, phones, or workstations. The promise is privacy, low latency, and reduced cloud cost. The risk is hardware marketing: The device is "AI-ready," but real use cases are few. The indicator of semantic emptying is the absence of useful local applications and real performance for relevant models.

Neuromorphic computing
The term has a research basis through architectures inspired by biological nervous systems and oriented toward energy efficiency. The risk is a vague overlap with AI. The indicator of semantic emptying is the use of "neuromorphic" without specific hardware, computing model, task, or demonstrated energy advantage.

6G
The term will inevitably have a hype period. It makes sense as a future generation of mobile communications but is vulnerable to including all possible promises: native AI, extreme latency, sensory networks, holographic communications, edge, and massive IoT. The indicator of semantic emptying is using it for products or strategies before standards, frequencies, use cases, and economic models are mature [52].

Digital public infrastructure
The term is important for digital governance, identity, payments, data, and public services. The risk is that it becomes a label for centralized government platforms without interoperability, democratic control, and

citizen protection. The indicator of semantic emptying is the absence of open standards, governance, equitable access, and audit mechanisms.

Data clean rooms

The term makes sense in advertising, retail, finance, and research, where organizations want to compare or analyze data without directly exposing them. The risk is presenting it as a complete privacy solution. The indicator of semantic emptying is the absence of an access model, control over queries, protection against reidentification, and audit.

Real-time enterprise

The term returns periodically. It promises organizations that react immediately to data and events. The risk is confusing rapidly updated data with better decisions. The indicator of semantic emptying is the absence of processes that can actually use real-time data.

These terms should not be rejected preventively. Some will remain, others will be absorbed into ordinary technical vocabulary, and others will become traces of a moment of commercial enthusiasm. The difference will be given by mechanism. The terms that survive will be those that can be tied to architectures, methods, data, risks, costs, and evaluation criteria. The others will continue, for a while, to function as signs of belonging to the present.

12 Conclusions

Terminological fashions in computer science are not mere accidents of vocabulary. They appear because the field evolves rapidly, and professional communities need terms that organize change. A new concept can name a real problem, concentrate resources, create standards, and accelerate better practices. Without such terms, innovation would remain scattered, difficult to communicate, and difficult to fund.

The difficulty appears after consecration. The term begins to circulate outside the technical community that gave it its initial meaning. It is taken over by management, marketing, consulting, funding programs, strategic documents, and public discourse. In this passage, the concept broadens. Sometimes it matures. Sometimes it empties. A technical term degrades when it can be used without a mechanism. A data warehouse becomes problematic when it no longer presupposes integration, history, quality, and analytical modeling. Cloud first becomes problematic when it no longer presupposes comparison among options, costs, and risks. Digital twin becomes problematic when it no longer presupposes a link with the physical system, updated data, and operational use. Zero Trust becomes problematic when it no longer presupposes identities, policies, segmentation, monitoring, and continuous verification. FinOps becomes problematic when it no longer presupposes cost visibility, distributed responsibility, and the relationship between cost and value. AI becomes problematic when it no longer presupposes models, data, evaluation, errors, and responsibility. At that point, the term no longer describes a solution. It merely signals the desire to belong to the present.

This observation does not justify rejecting new terms. A purely conservative attitude would be as weak as uncritical enthusiasm. Cloud computing changed software infrastructure. The data warehouse changed managerial reporting. DevOps changed application delivery. Machine learning produced real results in prediction, classification, and recommendation. Zero Trust corrected excessive trust in the network perimeter. FinOps introduced cost discipline into cloud architecture. PETs can enable more responsible use of sensitive data. Generative models are already changing intellectual work in many domains. The problem is not the term, but its use without criteria.

The analysis by families of promises shows that computer science periodically returns to the same aspirations. It wants to automate software development, turn data into knowledge, hide infrastructure, secure distributed systems, decentralize trust, reduce ecological impact, automate

processes, digitally double the physical world, reform economic sectors, and automate cognitive activities. Each family has produced useful instruments. Each has also produced excesses. This alternation follows from the tension among technical need, commercial pressure, and the desire of institutions to appear synchronized with novelty.

The practical effects can be costly. Terminological fashions can lead to excessive focus, disproportionate purchases, architectures imitated without necessity, pilots without continuation, overly ambitious centralized systems, premature automations, opportunistically renamed products, and security or sustainability strategies formulated more lexically than technically. Some projects fail not at implementation but at formulation: they have technology but no problem; a vendor but no user; a platform but no process; promised data but no real data; AI but no evaluation; Zero Trust but no policies; FinOps but no financial responsibility; and Green IT but no measurements.

For research, the risk is related. Fashionable terms can stimulate new topics but can also artificially homogenize scientific discourse. When every article becomes about AI, big data, smart cities, digital twins, quantum computing, FinTech, PETs, or digital transformation, even where the mechanism is absent, precision declines. The title becomes more current, but the contribution becomes less clear. Serious research must indicate exactly where novelty lies: in data, method, architecture, evaluation, context, or result. Lexical novelty is not a scientific contribution.

Minimal conceptual hygiene can reduce these risks. Terms must be locally defined, tied to mechanisms, compared with simpler solutions, and evaluated through verifiable criteria. A project invoking AI must explain what model it uses, what data it processes, and how quality is measured. A blockchain project must explain the trust problem it solves. A digital twin must indicate the physical system, data flow, model, and supported decision. A digital transformation must say what process changes. A Green IT project must measure consumption and ecological effect. A FinTech project must treat financial risk and regulation. A quantum computing project must indicate the problem, algorithm, and advantage over classical solutions.

The best question remains simple: what remains useful after the term loses its prestige? If cleaner data, clearer processes, better-organized code, easier-to-operate infrastructure, more visible costs, better security, real competences, or better decisions remain, the project can be justified. If nothing remains except temporary participation in a fashion, the term has functioned as a password, not as an instrument.

13 Limits of the analysis

This article does not offer a complete history of computer science. The selection of terms is oriented toward expressions that had high visibility in practice, management, applied research, and institutional discourse. Other concepts could be added: semantic search, knowledge graph, data fabric, composable enterprise, API economy, headless architecture, ambient computing, digital workplace, responsible AI, explainable AI, trustworthy AI, sovereign cloud, confidential computing, post-quantum cryptography, spatial computing. Some were omitted to preserve coherence. Others are too recent for it to be possible to say whether they will remain stable concepts or only transitional formulas.

The chronology is approximate. For many terms, the date of appearance is much older than their period of popularity. Artificial intelligence does not begin in 2022, machine learning does not begin in 2012, the idea of a software agent does not begin in 2024, quantum computing does not begin with recent commercial interest, and privacy-protecting techniques do not appear with the term PETs. The article follows the moment when terms become dominant in broad

professional discourse, not the moment of their first academic formulation.

The analysis does not claim that the terms discussed lack value. On the contrary, many produced important technical changes. The criticism concerns their use without mechanism, without criteria, and without relation to real need. This distinction must be preserved. Otherwise, the critique of fashion becomes a reverse fashion: the automatic rejection of novelty.

Another limit concerns cultural and institutional context. Terms do not circulate identically in universities, large companies, public administration, startups, or research. In a startup, FinTech may mean a concrete product tested on the market. In a public strategy, the same term may be a general formula. In a laboratory, quantum computing may mean serious fundamental research. In a commercial presentation, it may be only a sign of technological future. Analysis must be adapted to the place where the term is used.

14 Possible directions for extension

The analysis could be extended in several directions.

The first direction is bibliometric research. The frequency of terms such as "semantic web", "grid computing", "big data", "blockchain", "data mesh", "lakehouse", "digital twin", "metaverse", "generative AI", "agentic AI", "zero trust", "FinOps", "RPA", "hyperautomation," "quantum computing", or "privacy-enhancing technologies" could be followed in scientific databases by year, domain, and publication type. Such an analysis would show when the wave appears, when it reaches its peak, and when it begins to decline.

The second direction is the analysis of funding documents. Calls for projects, government strategies, and European programs frequently use current terms. It would be useful to see how these terms enter institutional language and how they modify the form of proposals. A plausible hypothesis is that certain expressions become almost mandatory in short periods, which leads to artificial convergence among projects.

The third direction is the analysis of failed projects. Not all failures are produced by hype, but many can be read through this lens. A project sometimes fails because the technology was chosen before the problem. At other times, failure appears because the term masked organizational complexity. A taxonomy of such failures would be useful for IT project management and for evaluating applied research proposals.

The fourth direction concerns differences among sectors. FinTech, HealthTech, EdTech, and GovTech use the same basic logic: the promise of reforming a domain through technology. Yet the risks are different. In FinTech there are financial and regulatory risks. In HealthTech, there are clinical risks. In EdTech there are pedagogical and privacy risks. In GovTech there are risks of access, equity, and administrative dependence.

The fifth direction concerns the difference between commercial hype and scientific maturity. Quantum computing, PETs, and AI provide good examples. All have a real technical basis. All can, however, be overstated. Research could follow how discourse changes when a concept moves from scientific articles to commercial presentations, public policies, and applied grants.

15 References

[1] W. H. Inmon, Building the Data Warehouse, 4th ed. Hoboken, NJ, USA: Wiley, 2005.

[2] R. Kimball and M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd ed. Hoboken, NJ, USA: Wiley, 2013.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009.

https://doi.org/10.1016/j.future.2008.12.001

[4] P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145. Gaithersburg, MD, USA: National Institute of Standards and Technology, 2011. https://csrc.nist.gov/pubs/sp/800/145/final

[5] National Institute of Standards and Technology, Zero Trust Architecture, NIST Special Publication 800-207. Gaithersburg, MD, USA: NIST, 2020. https://doi.org/10.6028/NIST.SP.800-207

[6] National Institute of Standards and Technology, "Privacy-Enhancing Cryptography," n.d. https://csrc.nist.gov/projects/pec

[7] OECD, "Privacy-enhancing technologies," n.d. https://www.oecd.org/en/topics/sub-issues/privacy-enhancing-technologies.html

[8] Royal Society, From Privacy to Partnership: The Role of Privacy Enhancing Technologies in Data Governance and Collaborative Analysis. London, U.K.: Royal Society, 2023. https://royalsociety.org/-/media/policy/projects/privacy-enhancing-technologies/from-privacy-to-partnership.pdf

[9] The World Bank, Fintech and the Future of Finance. Washington, DC, USA: World Bank, 2022. https://www.worldbank.org/en/publication/fintech-and-the-future-of-finance

[10] E. F. Codd, "A relational model of data for large shared data banks," Communications of the ACM, vol. 13, no. 6, pp. 377-387, 1970. https://doi.org/10.1145/362384.362685

[11] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, vol. 284, no. 5, pp. 34-43, 2001. https://www.scientificamerican.com/article/the-semantic-web/

[12] World Wide Web Consortium, "Semantic Web," n.d. https://www.w3.org/2001/sw/

[13] World Wide Web Consortium, "SPARQL Query Language for RDF," 2008. https://www.w3.org/TR/rdf-sparql-query/

[14] T. O'Reilly, "What is Web 2.0: Design patterns and business models for the next generation of software," O'Reilly Media, Sep. 30, 2005. https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html

[15] McKinsey & Company, "Common pitfalls in transformations: A conversation with Jon Garcia," n.d. https://www.mckinsey.com/capabilities/transformation/our-insights/common-pitfalls-in-transformations-a-conversation-with-jon-garcia

[16] Cloud Native Computing Foundation, "Observability," CNCF Cloud Native Glossary, 2024. https://glossary.cncf.io/observability/

[17] FinOps Foundation, "What is FinOps?" 2026. https://www.finops.org/introduction/what-is-finops/

[18] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in CIDR 2021, 2021. https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

[19] Z. Dehghani, "Data mesh principles and logical architecture," Martin Fowler, Dec. 3, 2020. https://martinfowler.com/articles/data-mesh-principles.html

[20] A. Goedegebuure, I. Kumara, S. Driessen, D. Di Nucci, G. Monsieur, W.-J. van den Heuvel, and D. A. Tamburri, "Data mesh: A systematic gray literature review," arXiv, 2023. https://arxiv.org/abs/2304.01062

[21] J. Preskill, "Quantum computing in the NISQ era and beyond," Quantum, vol. 2, art. 79, 2018. https://doi.org/10.22331/q-2018-08-06-79

[22] Gartner, "Gartner predicts over 40% of agentic AI projects will be canceled by end of 2027," Jun. 25, 2025. https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027
[23] C. Schmidt, "M. D. Anderson breaks with IBM Watson, raising questions about artificial intelligence in oncology," Journal of the National Cancer Institute, vol. 109, no. 5, art. djx113, 2017. https://doi.org/10.1093/jnci/djx113
[24] National Institute of Standards and Technology, "NIST releases first 3 finalized post-quantum encryption standards," Aug. 13, 2024. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards
[25] Gartner, "Gartner Hype Cycle identifies top AI innovations in 2025," Aug. 5, 2025. https://www.gartner.com/en/newsroom/press-releases/2025-08-05-gartner-hype-cycle-identifies-top-ai-innovations-in-2025
[26] ITPro, "One-size-fits-all agent governance sets enterprises up to fail," 2026. https://www.itpro.com/technology/artificial-intelligence/one-size-fits-all-agent-governance-sets-enterprises-up-to-fail
[27] National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework (AI RMF 1.0). Gaithersburg, MD, USA: NIST, 2023. https://doi.org/10.6028/NIST.AI.100-1
[28] National Institute of Standards and Technology, Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile. Gaithersburg, MD, USA: NIST, 2024. https://doi.org/10.6028/NIST.AI.600-1
[29] National Institute of Standards and Technology, "Post-Quantum Cryptography," n.d. https://csrc.nist.gov/projects/post-quantum-cryptography
[30] ITPro, "What is a vector database?" 2026. https://www.itpro.com/technology/big-data/what-is-a-vector-database
[31] Pinecone, "What is a vector database and how does it work?" May 3, 2023. https://www.pinecone.io/learn/vector-database/
[32] Anthropic, "Introducing the Model Context Protocol," Nov. 25, 2024. https://www.anthropic.com/news/model-context-protocol
[33] Model Context Protocol, "Specification," Mar. 26, 2025. https://modelcontextprotocol.io/specification/2025-03-26
[34] OpenAI, "Build your MCP server," OpenAI Developers, n.d. https://developers.openai.com/apps-sdk/build/mcp-server
[35] R. McKenna, G. Miklau, and D. Sheldon, "Winning the NIST Contest: A scalable and general approach to differentially private synthetic data," arXiv, 2021. https://arxiv.org/abs/2108.04978
[36] National Institute of Standards and Technology, "Differentially private synthetic data," May 3, 2021. https://www.nist.gov/blogs/cybersecurity-insights/differentially-private-synthetic-data
[37] National Institute of Standards and Technology, "Synthetic data generation," NIST Computer Security Resource Center Glossary, n.d. https://csrc.nist.gov/glossary/term/synthetic_data_generation
[38] Apple, "Introducing Apple Vision Pro: Apple's first spatial computer," Jun. 5, 2023. https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/
[39] G. Yenduri et al., "Spatial computing: Concept, applications, challenges and future directions," arXiv, 2024. https://arxiv.org/abs/2402.07912

[40] Google Design, "More human ambiance in ambient computing," n.d. https://design.google/library/mastering-ambiance-ambient-computing
[41] Samsung Semiconductor, "Ambient computing: Beyond ubiquitous," 2023. https://semiconductor.samsung.com/news-events/tech-blog/ambient-computing-beyond-ubiquitous/
[42] SAP, "What is a composable enterprise and who is the composer?" Dec. 22, 2022. https://community.sap.com/t5/additional-blog-posts-by-sap/what-is-a-composable-enterprise-and-who-is-the-composer/ba-p/13557568
[43] Gartner, "Gartner predicts more than 30% of the increase in demand for APIs will come from AI and tools using LLMs by 2026," Mar. 20, 2024. https://www.gartner.com/en/newsroom/press-releases/2024-03-20-gartner-predicts-more-than-30-percent-of-the-increase-in-demand-for-apis-will-come-from-ai-and-tools-using-llms-by-2026
[44] Gravitee, "The API economy: A deep dive in its definitions and best practices," Nov. 3, 2021. https://www.gravitee.io/blog/the-api-economy-a-deep-dive-in-its-definitions-and-best-practices
[45] Kong, "What is API economy?" n.d. https://konghq.com/learning-center/api-management/what-is-api-economy
[46] KPMG, "Embedded finance," n.d. https://kpmg.com/xx/en/our-insights/financial-services/embedded-finance.html
[47] D. Fett, P. Hosseyni, and R. Küsters, "An extensive formal security analysis of the OpenID Financial-grade API," arXiv, 2019. https://arxiv.org/abs/1901.11520
[48] OpenID Foundation, "Financial-grade API Security Profile 1.0 - Part 1: Baseline," Mar. 12, 2021. https://openid.net/specs/openid-financial-api-part-1-1_0.html
[49] The World Bank, "Digital financial services glossary," n.d. https://digitalfinance.worldbank.org/glossary
[50] Gartner, "Gartner unveils top emerging technologies to support autonomous business," Sep. 10, 2025. https://www.gartner.com/en/newsroom/press-releases/2025-09-10-gartner-unveils-top-emerging-technologies-to-support-autonomous-business
[51] Gartner, "Top strategic technology trends for 2026," 2026. https://www.gartner.com/en/articles/top-technology-trends-2026
[52] International Telecommunication Union, Framework and Overall Objectives of the Future Development of IMT for 2030 and Beyond, Recommendation ITU-R M.2160-0. Geneva, Switzerland: ITU, 2023. https://www.itu.int/rec/R-REC-M.2160-0-202311-I/en

Daniela TUDORICĂ graduated from the Faculty of Letters and Sciences of the Petroleum-Gas University of Ploiești, in the specialty Mathematics and Informatics, in 2002. She obtained a master's degree in Informatics in 2008 and a second master's degree in Advanced Automations and Programmable Structures in 2009, both from the Petroleum-Gas University of Ploiești. She got the title of doctor in systems engineering in 2014, with the thesis "Contributions Regarding the Automatic Monitoring of Pipeline Transport Systems for Petroleum Products". At present she is a lecturer in the Department of Informatics, Information Technology, Mathematics and Physics of the Petroleum-Gas University of Ploiești. Her teaching and research activity includes programming fundamentals, advanced programming techniques, numerical calculus, computational geometry, modelling and simulation, advanced data mining techniques and operations research. Her domains of work are: computer

programming, applied informatics, numerical methods, statistics, data mining, modelling and simulation, and the automatic monitoring of technical systems. She is author or co-author of 6 books and has published 13 articles in journals indexed in international databases or in the proceedings of international scientific events.

Antonia PATRASCU graduated from the Faculty of Economic Sciences at the Petroleum-Gas University of Ploiești, completed a Bachelor's degree programme in Economic Informatics, in 2020. She is a second-year PhD student at the Doctoral School of Economic Sciences "Eugeniu Carada", University of Craiova, in the field of Economic Informatics. She is currently a Assistant Lecturer at the Petroleum-Gas University of Ploiești, Faculty of Economic Sciences. She teaches laboratory activities for the following courses: Databases, Computer Programming, Digital Technologies in Economics, Advanced IT Technologies, Internet Technologies, Computer Graphics and Programming, and Electronic Business.

Bogdan George TUDORICĂ graduated from the Petroleum-Gas University of Ploiești, Faculty of Letters and Sciences, with a degree in Mathematics and Informatics, in 1998. He obtained a master's degree in Databases as Business Support from the Bucharest University of Economic Studies in 2009 and received the title of doctor in economic informatics in 2014, with the thesis "Solutions for Organizing Large Data Volumes". In 2024 he obtained the habilitation, with the thesis "From Storage to Understanding: The Evolution of Data Analysis and Databases in the Age of Artificial Intelligence". At present he is a professor at the Petroleum-Gas University of Ploiești, Faculty of Economic Sciences, and director of the Department of Cybernetics, Economic Informatics, Finance and Accounting. His domains of work are: computer programming, computer networks and data communications, databases, data analysis, information security, graphics and multimedia, Internet and web technologies, and the use of computers in education. He has also been active as a trainer, Cisco instructor, Oracle instructor and ECDL examiner. His professional activity includes the design and implementation of e-learning platforms, educational web resources and academic websites, as well as participation in research and development projects related to digital education, business informatics, energy efficiency, artificial intelligence and smart technologies.