International collective monograph

SCIENCE AND EDUCATION IN THE THIRD MILLENNIUM:

Information Technology, Education, Law, Psychology, Social Security And Work, Management

Volume I

LUBLIN, POLSKA
2026

INSTITUTE OF PUBLIC ADMINISTRATION AFFAIRS

SCIENCE AND EDUCATION IN THE THIRD MILLENNIUM:

Information Technology, Education, Law, Psychology, Social Security And Work, Management

international collective monograph

Volume I

Lublin, Polska

2026

UDC 001.9
C 24
https://doi.org.10.5281/zenodo.....

Recommended for publication by INSTITUTE OF PUBLIC ADMINISTRATION AFFAIRS № 5 dated 2026-06-06

Authors: Tran Minh DUC, A. DEMIANIUK, H. DYMOVA**,** K. KALIUGA, N. KOLOMYYCHUK, S. SAVCHUKO, O. KOROTUN, V. KOSOVA, T. LOZOVA, S. NYKYPORETS, Y. BOIKO, O. OHIRKO, M. RIABENKA, V. POSTOVA, O. ROMANIUK, A. STAKHOV, O. SHAMRUK, V. HRECHANYI, V. SYDORCHUK, V. SOLODKA, M. PATLAIENKO, O. TSYSELSKA, V. VASIUTA, V. VASIUTA, Y. VYKHODETS, T. YAKOVYSHYNA,

SCIENCE AND EDUCATION IN THE THIRD MILLENNIUM: Information Technology, Education, Law, Psychology, Social Security and Work, Management. *International collective monograph. Volume I.* Institute of Public Administration Affairs. Lublin, Polska, 2026. 764 p.

ISBN: 978-83-68095-80-9
ISBN: 978-617-95591-1-2

The collective monograph is the result of the generalization of the conceptual work of scientists who consider current topics from such fields of knowledge as: pedagogy, history, computer science, (AI) artificial intelligence, law, physical culture, through the prism of interactive technologies in education.

For scientists, educational staff, PhD candidates, masters of educational institutions, university faculties, stakeholders, managers and employees of management bodies at various hierarchical levels and for everyone, who is interested in current problems of pedagogy, history, computer science, (AI) artificial intelligence, law, physical culture, through the prism of interactive technologies in education.

AUTHORS:

CHAPTER 1.
Tran Minh DUC
Faculty of Cultural Industries,
Thu Dau Mot University, Vietnam
ductm@tdmu.edu.vn
https://orcid.org/0009-0003-9359-3000

CHAPTER 2.
Antonina DEMIANIUK
PhD in Economics, Associate Professor
West Ukrainian National University,
Department of Educology and Pedagogy,
11, Lvivska Str. (WUNU Building 1),
Ternopil, 46009, Ukraine
antonina_demyanyuk@ukr.net
https://orcid.org/0000-0002-8994-7633

CHAPTER 3.
Hanna DYMOVA
Candidate of Technical Sciences, PhD.,
Associate Professor, Department of
Management, Marketing and Information
Technology Kherson State Agrarian and
Economic University (73006, 23 Stritenska
st., Kherson, Ukraine; 25031, 5/2
University ave., Kropyvnytskyi,
Kirovohrad region, Ukraine)
dymova_g@ksaeu.kherson.ua
https://orcid.org/0000-0002-5294-1756

CHAPTER 4.
Karina KALIUGA
Doctor of Law, Action Head of the
Department
of criminal law, process and criminalistics
Institute of Economics and Law of the
Classic Private University, Professor,
(Zaporizhzhya, Ukraine)
monkeywise7@gmail.com
https://orcid.org/0000-0001-8247-2324

CHAPTER 5.
Nataliya KOLOMYYCHUK
PhD in Economics, Associate Professor,
Associate Professor of the Department of
Finance named after S. I. Yuriy,
West Ukrainian National University,
Ternopil, Ukraine
Kolomyychuk.natalya@ukr.net
https://orcid.org/0000-0001-6160-9053

Svitlana SAVCHUK
PhD in Economics, Associate Professor,
Associate Professor of the Department of
Finance named after S. I. Yuriy,
West Ukrainian National University,
Ternopil, Ukraine
sv_savchuk@ukr.net
https://orcid.org/0000-0002-6213-4122

CHAPTER 6.
Olha KOROTUN
PhD in Economics, Associate Professor,
Associate Professor of the Department of
Marketing, The National University of
Water and Environmental Engineering
(33028, Ukraine, Rivne, Soborna St., 11)
o.p.korotun@nuwm.edu.ua
https://orcid.org/0000-0002-5628-8301

AUTHORS:

CHAPTER 7.
Vera KOSOVA
Assistant, Department of Biotechnology and Engineering National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
vera_62@ukr.net
https://orcid.org/0000-0002-3441-6536

CHAPTER 8.
Tetiana LOZOVA
PhD (Economics), Associate Professor, Research Associate of the Department of Integration of Science, Education and Business,
State Organization «Institute of Market and Economic-Ecological Researches of NAS of Ukraine, (29 Frantsuzkyi Blvd., Odesa, 65044, Ukraine)
ua.lozovaya@gmail.com
https://orcid.org/0000-0002-4896-1930

CHAPTER 9.
Svitlana NYKYPORETS
senior English language lecturer Vinnytsia National Technical University, Department of Foreign Languages (95, Khmelnytske shose, Vinnytsia, 21021)
fotinia606@gmail.com
https://orcid.org/0000-0002-3546-1734

Yuliia BOIKO
senior English language lecturer Vinnytsia National Technical University, Department of Foreign Languages, (95, Khmelnytske shose, Vinnytsia, 21021, Ukraine)
julivasb@gmail.com
https://orcid.org/0009-0003-7877-1921

CHAPTER 10.
Oleh OHIRKO
Doctor of Philosophy, Associate Professor, Associate Professor of the Department of Philosophy and Pedagogy of the Stepan Gzhytskyi National University of Veterinary Medicine and Biotechnologies of Lviv, member of the public organization "International Educators and Scholars Foundation, IESF" (79010, 50 Pekarska street, Lviv, Ukraine)
OhirkoOleh@gmail.com
https://orcid.org/000-0001-7563-6992

AUTHORS:

CHAPTER 11.
Maryna RIABENKA
PhD in Economics, Associate Professor, Associate Professor of the Department of Tourism and Hotel and Restaurant Business Vinnytsia Institute of Trade and Economics of State University of Trade and Economics (Soborna, 87, 21000, Vinnytsia, Ukraine)
m.riabenka@vtei.edu.ua
https://orcid.org/0000-0002-3024-137X

Valentyna POSTOVA
PhD in Economics, Associate Professor, Associate Professor of the Departmen of Tourism and Hotel and Restaurant Business Vinnytsia Institute of Trade and Economics of State University of Trade and Economics (Soborna, 87, 21000, Vinnytsia, Ukraine)
v.postova@vtei.edu.ua
https://orcid.org/0000-0002-0056-5648

CHAPTER 12.
Oleksandr ROMANIUK
Doctor of Technical Sciences, Professor, Vinnytsia National Technical University,
rom8591@gmail.com ,
https://orcid.org/0000-0002-2245-3364

Alexey STAKHOV
Candidate of Technical Sciences, Senior Lecturer, Vinnytsia National Technical University,
aleksey.stahov@gmail.com ,
https://orcid.org/0000-0002-4901-3211

CHAPTER 13.
Oleh SHAMRUK
Candidate of Psychological Sciences, Head of the Department of Professional Military Training, Centre for Postgraduate Education, Kyiv Institute of the National Guard of Ukraine, Kyiv, Ukraine
shamrukoleg@gmail.com
https://orcid.org/0009-0008-6138-5318

Viacheslav HRECHANYI
Lecturer at the Department of Professional Military Training, Center for Postgraduate Education, Kyiv Institute of the National Guard of Ukraine
gre4anyy.v@gmail.com
https://orcid.org/0009-0009-8480-4172

Viktoriia SYDORCHUK
Senior Lecturer, Professional Military Training, Center for Postgraduate Education, Kyiv Institute of the National Guard of Ukraine
vitaprimak11@gmail.com
https://orcid.org/0000-0002-2962-1113

CHAPTER 14.
Valentyna SOLODKA
PhD in Technical Sciences, Associate Professor, State University of Intellectual Technologies and Communications,
(1 Kuznechna Street, Odesa, 65029, Ukraine)
valyaonas@gmail.com
https://orcid.org/0000-0002-5357-6682

Mykola PATLAIENKO
PhD in Technical Sciences, Associate Professor, State University of Intellectual Technologies and Communications,
(1 Kuznechna Street, Odesa, 65029, Ukraine)
nick_msa@ukr.net
https://orcid.org/0000-0002-5565-8956

AUTHORS:

CHAPTER 15.
Oksana TSYSELSKA
Candidate of Art Studies, Senior Lecturer, Department of directing and acting, Kyiv National University of Culture and Arts
oksanatsyselska@gmail.com
https://orcid.org/0000-0002-2869-2454

CHAPTER 16.
Viktoriia VASIUTA
PhD in Technical, Associate Professor, Associate Professor of Department of Economics, Entrepreneurship and Marketing, National University «Yuri Kondratyuk Poltava Polytechnic», (24, Vitaliia Hrytsaienka, Poltava, 36011, Ukraine)
Vasuta_V_B@meta.ua
https://orcid.org/0000-0002-7469-3968

Vasyl VASIUTA
PhD in Technical, Associate Professor, Associate Professor of the Department of Computer and Information Technologies and Systems, National University «Yuri Kondratyuk Poltava Polytechnic», (24, Vitaliia Hrytsaienka, Poltava, 36011, Ukraine)
Vasuta_V_V@ukr.net
https://orcid.org/0000-0001-6209-1129

CHAPTER 17.
Yuriy VYKHODETS
Doctor of Philosophy, Associate Professor, Professor of the Department of Criminology and Information Technologies, National Academy of Internal Affairs
orcid.org/0009-0008-1312-9248

CHAPTER 18.
Tetiana YAKOVYSHYNA
Candidate of Pedagogical Sciences, Associate Professor, Department of Primary, Inclusive and Higher Education Pedagogy Rivne State University of t Humanities (Plastova str., 31, Rivne, 33000, Ukraine)
yakovyshynat@gmail.com
https://orcid.org/0000-0001-9507-3621

DOI 10.5281/zenodo.20810460
UDC 81'23:81'373.612.2:004

CHAPTER 9.
ANTHROPOMORPHIC METAPHORS IN CONTEMPORARY TECHNICAL DISCOURSE: A COGNITIVE-LINGUISTIC ANALYSIS OF THE IT TERMINOLOGICAL SYSTEM

Svitlana NYKYPORETS
senior English language lecturer
Vinnytsia National Technical University, Department of Foreign Languages,
(95, Khmelnytske shose, Vinnytsia, 21021, Ukraine)
fotinia606@gmail.com
https://orcid.org/0000-0002-3546-1734

Yuliia BOIKO
senior English language lecturer
Vinnytsia National Technical University, Department of Foreign Languages,
(95, Khmelnytske shose, Vinnytsia, 21021, Ukraine)
julivasb@gmail.com
https://orcid.org/0009-0003-7877-1921

Abstract. This work examines anthropomorphic metaphors as a structured cognitive mechanism in contemporary IT discourse rather than as a purely stylistic feature. The study addresses the problem of how abstract computational phenomena are conceptualised through human-centred models of body, behaviour, cognition, social hierarchy, health and agency. Drawing on cognitive metaphor theory and conceptual integration theory, the chapter analyses IT terms such as *computer virus*, *zombie process*, *parent/child component*, *smart agent*, *memory leak*, *daemon* and *client-server architecture*. The findings show that anthropomorphic metaphorisation is organised around three dominant microsystems: biological-physiological, psychological-intellectual and social-hierarchical. These metaphorical models reduce abstraction, support professional communication and facilitate the onboarding of new developers through familiar cognitive frames. The chapter also demonstrates that anthropomorphic terminology has significant pragmatic value in user-oriented communication, technical documentation, debugging discourse and interface design. At the same time, the study identifies potential risks of over-anthropomorphising artificial intelligence systems, especially when terms such as *learning*, *reasoning* or *hallucination* are interpreted too literally. The results are relevant for technical writers, translators and localisation specialists, as they show the need to preserve both conceptual accuracy and communicative accessibility in IT terminology.

Keywords: conceptual mapping, lexical semantics, human-computer interaction, semantic extension, software nomenclature.

Introduction. The unprecedented proliferation of information technologies in the twenty-first century has generated an equally remarkable expansion of specialist vocabulary. From the architectures of large language models to the protocols governing distributed systems, and from the threat taxonomies of contemporary cybersecurity to the abstractions of cloud-native engineering, the information technology (IT) sector continually demands new nominative units capable of representing concepts for which no prior lexical tradition exists. This terminological pressure is particularly acute in the current era, characterised by the rapid emergence of generative artificial intelligence, microservice orchestration, and adversarial machine learning — domains in which novel entities and processes simultaneously require technical precision and broad cognitive accessibility.

Among the most productive mechanisms of term formation operative in this domain is *anthropomorphic metaphor*: the systematic conceptual transfer of attributes, behaviours, or relational structures belonging to the human sphere onto the entities and processes of the digital environment. Widely attested examples – *parent* and *child* processes, the *daemon* background service, *memory leak*, *computer virus*, *smart agent*, *dead letter queue* – testify to a recurrent cognitive strategy rather than an accidental accumulation of figurative usages. This strategy reflects what cognitive science has long identified as a fundamental feature of the human mind: the disposition to comprehend abstract or unfamiliar domains through the conceptual lens of familiar, embodied experience *(Lakoff & Johnson, 2008)*.

The present monograph is grounded in Conceptual Metaphor Theory (CMT), as elaborated by George Lakoff and Mark Johnson in their foundational work *Metaphors We Live By (1980)*, which established that metaphor is not a peripheral ornament of language but a central mechanism of human cognition, structuring the way in which abstract experience is both understood and communicated. Within this framework, a conceptual metaphor such as A COMPUTER IS A PERSON encodes a systematic mapping from a source domain – the human being, with its biological processes, social

relations, and psychological states – onto a target domain comprising digital artefacts and computational processes.

This theoretical model is complemented by the Theory of Conceptual Integration (Blending Theory), developed by Gilles Fauconnier and Mark Turner (*2002*), which accounts for the dynamic, on-line construction of meaning through the partial blending of distinct mental spaces. Where CMT illuminates structural mappings between *(Kot et al., 2025)* stable cognitive domains, blending theory captures the emergent meanings arising in actual terminological coinages and professional discourse. Together, these two frameworks provide a sufficiently nuanced theoretical apparatus for analysing IT anthropomorphisms at both the level of systematic cross-domain mapping and the level of individual terminological innovation. Technical discourse is understood throughout this study as a dynamic semiotic system encoding the professional world-view of software engineers, developers, and system architects *(Geeraerts & Cuyckens, 2007)*. IT terminology is accordingly not a neutral repository of denotative labels but a cognitively and socially loaded linguistic practice that reveals how practitioners conceptualise the digital environments they design and inhabit.

Despite the pervasive presence of anthropomorphic expressions in both official IT standards – including API documentation, Request for Comments (RFC) specifications, and operating system manuals – and in the informal registers of developer communities such as GitHub and Stack Overflow, the existing linguistic literature offers no comprehensive cognitive classification of anthropomorphic metaphors specific to contemporary computer terminology. Studies of metaphor in scientific and technical language have tended to concentrate on long-established fields such as medicine or physics; the IT terminological system – in both its standardised and informal dimensions – has received comparatively limited systematic attention from a cognitive-linguistic perspective. This lacuna is particularly significant given the rapidity with which new anthropomorphic coinages are entering the lexicons of generative AI engineering, distributed computing, and cybersecurity, thereby reshaping the conceptual infrastructure of the discipline.

The present study pursues the following principal objectives: (i) to identify and catalogue the key anthropomorphic metaphors operative in contemporary IT discourse, drawing on professional dictionaries, technical specifications, and live developer corpora; (ii) to describe the cognitive matrices and vectors of knowledge transfer from the donor domain (HUMAN BEING) to the target domain (COMPUTING TECHNOLOGIES), with particular attention to biological, social-relational, behavioural, and psychological sub-mappings; and (iii) to assess the terminological and epistemological implications of anthropomorphisation for the professional conceptualisation of digital systems.

The study employs a corpus-based methodology. Source material is drawn from professional IT glossaries, RFC standards, open-source API documentation, and developer discourse extracted from GitHub repositories and Stack Overflow threads. Terminological units are subjected to cognitive-discursive analysis and componential semantic analysis so as to elucidate their structural and relational properties. The initial retrieval and preliminary classification of bibliographic sources and terminological examples were supported by Claude Sonnet 4.6 *(Anthropic, 2025)*, an AI language model employed as an analytical instrument for systematic literature search and data pre-processing; all findings were subsequently verified and interpreted through independent scholarly review in accordance with standard philological methodology.

Subsequent analytical procedures were executed via the *Voyant Tools* platform. This open-access, web-based suite facilitates the examination of either uploaded textual data or documents sourced from designated URLs through an assortment of analytical instruments. This computational input-output pipeline converts intricate metadata into highly legible, intuitive visualisations. Presently available without financial barriers, the infrastructure demands merely an active internet connection and a specific textual dataset (namely, a corpus). Consequently, researchers possessing diverse levels of expertise and technical proficiency can leverage the platform to extract salient patterns inherent within their source material. The comprehensive *Voyant Tools* architecture

encompasses a variety of applications, including *TextualArc*, *Links*, *TermsBerry*, *Collocates*, *Correlations*, *WordTree*, *Phrases*, *Topics*, *Contexts*, *Mandala*, *Terms*, *Microsearch*, *Dreamscape*, *RezoViz*, and *Reader*, among others.

In the context of the present inquiry, empirical focus was directed towards the *TermsBerry* tool, which enables the scrutiny of high-frequency lexical items alongside their respective collocates (words appearing in close textual proximity). The primary utility of this specific instrument lies in its capacity to amalgamate the visualisation of recurrent terminology with a rigorous exploration of lexical co-occurrence patterns, thereby mapping the precise degree to which these terms appear adjacent to one another within the corpus.

1. Cognitive mechanisms and donor-acceptor zones of anthropomorphisation in the IT domain.

The prevailing assumption in pre-cognitive linguistics held that metaphor was essentially an ornamental feature of literary language – a stylistic deviation from a presumed neutral, literal baseline. This view was fundamentally challenged by Lakoff and Johnson *(2008)*, who demonstrated that metaphor is not peripheral to language but constitutive of thought itself. Within Conceptual Metaphor Theory, metaphorical mappings are understood as systematic correspondences between two conceptual domains: a relatively familiar and experientially concrete *source domain* and a comparatively abstract or less accessible *target domain*. The cognitive work performed by such mappings is not decorative but epistemic – they provide the conceptual scaffolding through which unfamiliar phenomena are apprehended, categorised, and communicated.

In the domain of information technology, this epistemic function is of particular salience. The entities and processes that populate digital environments – threads, stacks, memory spaces, network topologies – are in principle invisible, operating at levels of abstraction far removed from ordinary sensorimotor experience. The human cognitive apparatus, however, is fundamentally shaped by embodied interaction with a physical and social world *(Lakoff & Johnson, 1980)*. Accordingly, the conceptualisation of computational systems through the prism of human anatomy, behaviour, and social

organisation constitutes not a rhetorical convenience but a cognitive necessity: a means of rendering tractable a domain that would otherwise resist intuitive comprehension.

This perspective aligns with the broader programme of *embodied cognition*, which holds that conceptual structures are grounded in bodily schemas derived from the organism's sensorimotor engagement with its environment. Abstract concepts – including the elaborate abstractions of computer science – are not processed as purely formal, disembodied symbols; rather, they are stabilised and made communicable through their mapping onto experientially anchored image schemas such as CONTAINMENT, PATH, or SOURCE-PATH-GOAL. Anthropomorphic metaphor in IT terminology represents a particularly rich instantiation of this grounding strategy, one in which the source domain is not merely the physical body in isolation but the full range of human agency, social life, and biological organisation.

The principal conceptual vector operative in IT anthropomorphic metaphor may be formalised as Human Being (Source Domain) → Computational System (Target Domain). This mapping is not a single, monolithic correspondence but a structured network of partial projections, each of which selects certain features of the source domain and maps them onto analogous structural positions within the target. The mapping is partial – that is, not all properties of human beings are transferred to computational systems – and asymmetrical: it is the familiar, experientially grounded domain that lends its conceptual structure to the abstract domain, not vice versa.

The cognitive motivation for this directional transfer is readily apparent. Software architectures, communication protocols, *(Sachaniuk-Kavets'ka et al., 2026)* and operating system processes are highly abstract constructs whose internal organisation and causal dynamics are not directly perceptible. By projecting human anatomical structure, social roles, and intentional behaviour onto these systems, practitioners create conceptual handles that facilitate both individual cognition and collective communication within professional communities. A developer who says that a process *listens* on a port, or that a server *responds* to a request, is not engaging in deliberate metaphorical embellishment;

rather, they are exploiting an entrenched cross-domain mapping that renders the functional behaviour of the system immediately intelligible in terms of human communicative action.

Fauconnier and Turner's (*2002*) theory of conceptual integration extends this analysis by accounting for the dynamic, compositional dimension of terminological meaning construction. Rather than straightforwardly projecting one domain onto another, the mind constructs *blended spaces* in which elements from both the human and the computational domain are selectively integrated, yielding emergent conceptual structures that belong to neither source nor target alone. The notion of a *daemon* – a background process in Unix-like operating systems – exemplifies such blending: it combines the human-domain concept of an invisible, autonomous agent performing ceaseless unseen work with the computational concept of a persistent, non-interactive process, thereby producing a stable conceptual blend that has become thoroughly lexicalised within the terminological system.

It is equally important to note that conceptual transfer in IT anthropomorphism is not exclusively unidirectional. Whilst the dominant vector runs from the human to the computational sphere, a secondary, reflexive vector has become increasingly operative (*Stepanova et at., 2025*) in contemporary discourse: terms coined originally to anthropomorphise digital systems – *neural*, *agent*, *intelligence*, *memory* – have subsequently re-entered general and scientific language carrying computational connotations that enrich, and occasionally displace, their prior human-centred meanings. This bidirectional dynamic underscores the degree to which IT terminology is not a passive recipient of human conceptual frames but an active participant in the ongoing reshaping of the broader cognitive lexicon.

The cross-domain mappings identified above are not distributed randomly across the IT lexicon but cluster into recognisable *semantic zones*, each corresponding to a coherent sub-domain of the human source. The present study identifies two foundational zones of particular analytical importance for the purposes of this subchapter; further zones – covering kinship and social relations, pathological states, and affective processes – are examined in detail in subsequent sections.

The anatomical-somatic zone encompasses terminological units in which computational entities are assigned structural correlates of the human or animal body. Illustrative examples include *bus* – the shared communicative conduit whose name maps the image of a vehicle route carrying passengers onto the concept of a shared data pathway – *backbone* (the central high-capacity transmission network, mapped from the vertebral column as the organism's primary structural axis), *memory* (the system's capacity to store and retrieve data, projected from the human faculty of retention and recall), and *head* (the read/write element of a magnetic storage device, positioned at the proximal apex of the assembly and construed as the primary sensing organ of the hardware body). What these mappings share is the attribution of organic structural integrity to artefacts: the computational system is implicitly construed as a body with differentiated, specialised components standing in relations of functional complementarity, much as the organs of a living organism sustain and depend upon one another.

The behavioural zone is defined by the projection of intentional, volitional, or agentive human action onto software processes and network entities. Within this zone, programs and services are attributed the capacity to act independently and purposively: a service *listens* for incoming requests, a thread *sleeps* or *wakes*, a process *dies* or *crashes*, a function *invokes* a callback, a client *handshakes* with a server. The linguistic forms exploited here are predominantly verbs of human agency, sensory perception, or biological cycle. Their systematic application to computational processes implies a model of the software environment as populated by quasi-autonomous actors whose behaviour is intelligible in terms of human intentional action – a model that, as subsequent analysis will demonstrate, extends well beyond individual lexical items to inform the entire architectural metaphorology of contemporary operating systems and distributed computing frameworks.

2. Structural-semantic taxonomy of anthropomorphic IT terms.

The structural-semantic taxonomy of anthropomorphic IT terms demonstrates that the metaphorical "humanisation" of computational systems is not chaotic. It is

organised around recurrent conceptual domains that allow specialists to interpret technical entities as if they possessed life, cognition, social roles, hierarchical relations or behavioural capacities. In contemporary (*Nykyporets et al., 2024*) IT discourse, anthropomorphic terminology may be grouped into three broad microsystems: the biological-physiological sphere, the psychological-intellectual sphere and the social-hierarchical sphere. These microsystems are not fully isolated from one another. A single term may activate several semantic zones simultaneously. For example, zombie process belongs to the biological-physiological sphere because it refers to an abnormal post-mortem state, but it also belongs to the behavioural sphere because it denotes a process that remains present in the system despite having completed execution. Similarly, trusted agent combines a social metaphor of delegated agency with an epistemic metaphor of trust.

The first microsystem, the biological-physiological sphere, represents computational systems as quasi-living entities. In this model, systems are born, run, grow, become infected, suffer from vulnerabilities, leak resources, die, recover or pass through a lifecycle. The metaphorical source domain is the living organism, while the target domain includes processes, files, networks, software products, data objects and security environments. This microsystem is especially productive in cybersecurity and system administration, where the language of health, disease and death provides an efficient framework for describing damage, malfunction, persistence and recovery. Terms such as computer virus, worm, infection, quarantine, hygiene, kill a process, zombie process, orphan process, heartbeat signal and data lifecycle show that computational entities are conceptualised through patterns of biological existence and physiological functioning.

The term computer virus is one of the most conventionalised examples of biological metaphor in IT terminology. It frames malicious code as an infectious organism that enters a host, replicates and damages the system. This metaphor is effective because it transfers an already familiar medical model into the digital domain: a computer becomes a host, malicious software becomes a pathogen, antivirus tools

become protective or therapeutic mechanisms, and cybersecurity practices become hygiene. The same logic underlies terms such as worm, infection, quarantine and malware outbreak. These terms are not simply figurative. They structure professional reasoning about threat propagation, containment and remediation. The metaphor helps cybersecurity specialists *(Ibrahimova et al., 2025)* speak about digital risk in terms of contamination, immunity, exposure and treatment.

The life-and-death model is also central to process management. A process may be created, spawned, suspended, killed or terminated. In Unix-like systems and many programming contexts, to kill a process means to send a signal that causes it to stop. The metaphor is stark, but technically economical. It conceptualises forced termination as an act of ending a process's operational life. Zombie process extends the same frame. It denotes a process that has completed execution but still retains an entry in the process table because its parent has not collected its exit status. The process is "dead" in terms of execution, but it remains visible in the system. Orphan process similarly depends on a family and lifecycle metaphor: it refers to a process whose parent has terminated before it. Here biological and social metaphors intersect, since the process is conceptualised both as a living entity and as a member of a family structure.

The term data lifecycle demonstrates a softer biological model. It does not represent data as a living organism in a literal sense, but it organises data management through stages analogous to life: creation, storage, use, sharing, archiving and deletion. This metaphor is important (*Stepanova et al., 2025*) in information governance, data protection and enterprise architecture because it transforms data into an entity with a temporal trajectory. Data is not merely stored; it is born, used, maintained, retired and eventually destroyed. In this sense, lifecycle terminology provides a procedural and regulatory framework for handling digital objects.

Table 1.

Biological-physiological metaphors in IT terminology: source-domain elements and cognitive functions

Microsystem	Representative IT terms	Source-domain elements	Target-domain interpretation	Cognitive function
Biological-physiological	*computer virus, worm, infection, malware outbreak*	pathogen, contagion, disease spread	malicious code propagation across systems	Explains cybersecurity threats through epidemiological reasoning
	quarantine, antivirus, hygiene, patching hygiene	isolation, treatment, prevention, cleanliness	containment and prevention of digital threats	Frames security as health maintenance
	kill a process, terminate, deadlock, zombie process	death, abnormal survival, bodily failure	process termination or abnormal execution state	Makes system states easier to diagnose
	orphan process, child process, parent process	family loss, descent, dependency	process hierarchy and lifecycle dependency	Explains control relations through kinship
	heartbeat, watchdog, health check	pulse, monitoring, survival	service availability and system monitoring	Conceptualises reliability as vitality
	data lifecycle, software lifecycle, end of life	birth, growth, ageing, death	stages of data or software existence	Structures governance and maintenance processes
	memory leak, resource starvation	bodily leakage, deprivation	uncontrolled resource loss or insufficient allocation	Represents performance degradation as physiological damage
	healing, self-healing system, recovery	repair, restoration, convalescence	automated restoration after failure	Frames resilience as bodily recovery

Source: created by authors on the bases of present research.

The second microsystem is the psychological-intellectual sphere. It includes terms that attribute cognitive, perceptual or quasi-emotional capacities to computational systems. This group has become particularly salient in the era of artificial intelligence, machine learning and autonomous software agents. Terms such as *smart contract, deep learning, neural network, machine learning, attention mechanism, memory leak, blind spot, hallucination, inference engine, reasoning model, intelligent agent and knowledge base* represent technical systems as entities capable of learning, remembering, perceiving,

reasoning or making decisions. Although such terms often have highly specific mathematical or engineering meanings, their linguistic form draws heavily on human cognition.

The term neural network is a central example. It maps the structure of biological neural systems onto computational architectures consisting of interconnected nodes or artificial neurons. The metaphor does not imply that an artificial neural network operates identically to the human brain. Rather, it transfers selected features: distributed processing, weighted connections, pattern recognition and adaptation. Similarly, deep learning uses the spatial metaphor of depth to refer to multiple layers in a neural architecture, but it also evokes the human idea of advanced or profound learning. The term is therefore both technical and metaphorically loaded. It frames machine optimisation as a form of learning, even though the process is based on statistical adjustment rather than human understanding.

Smart contract is another significant example. In blockchain discourse, a smart contract is not smart in the human sense and is not necessarily a contract in the traditional legal sense. The term nevertheless combines two anthropocentric frames: intelligence and social agreement. It suggests an automated arrangement that can execute predefined conditions without direct human intervention. The adjective smart attributes instrumental rationality to code, while contract activates the social-legal model of obligation and agreement. The term is powerful because it compresses a complex technical and institutional mechanism into a cognitively accessible expression.

The term blind spot shows how perceptual metaphors are transferred into technical discourse. In human experience, a blind spot is a zone that cannot be seen, either physiologically or metaphorically. In cybersecurity, data analysis, AI evaluation or system monitoring, a blind spot refers to an area that remains undetected, unmonitored or poorly understood. The metaphor is effective because it connects technical invisibility with bodily limitation. It also has a critical function: it indicates that a system's failure may result not from the absence of data, but from the inability to perceive or interpret relevant signals.

The expansion of generative AI has intensified the use of psychological and intellectual metaphors. Terms such as hallucination, reasoning, memory, alignment,

attention and agentic behaviour are now common in professional and public discussions of AI. These metaphors are productive but potentially risky. They support explanation by using familiar cognitive categories, but they may also overstate the extent to which computational systems possess human-like mental states. For instance, hallucination in generative AI usually refers to the production of plausible but false or unsupported output. The term is useful because it conveys a deviation from reliable representation, but it also borrows from a human psychiatric and perceptual frame. This may make the phenomenon easier to recognise, while also suggesting a stronger analogy with human experience than the underlying mechanism justifies.

Table 2.

Psychological-intellectual metaphors in IT terminology: human-centred source models, technical meanings and risks of overextension

Microsystem	Representative IT terms	Human-centred source model	Technical target meaning	Possible risk of overextension
Psychological-intellectual	*neural network, artificial neuron*	brain structure, neural connectivity	computational model with interconnected units	May imply excessive similarity to biological cognition
	deep learning, machine learning	learning, skill acquisition, adaptation	statistical optimisation from data	May obscure the non-human nature of model training
	smart contract, smart device, smart agent	intelligence, practical rationality	automated or rule-based technical behaviour	May exaggerate autonomy or judgement
	attention mechanism	human attention and focus	weighting of input features in a model	May imply conscious focus
	memory, memory leak, cache memory	remembering, retaining, forgetting	data storage and retrieval mechanisms	May blur storage with cognitive memory
	blind spot, visibility gap	perceptual limitation	unmonitored or undetected technical area	May simplify complex observability failures
	hallucination	distorted perception or false experience	false or unsupported generative AI output	May anthropomorphise statistical text generation
	inference engine, reasoning model	logical thought, conclusion-making	rule-based or model-based output generation	May imply genuine understanding

Source: created by authors on the bases of present research.

The third microsystem is the social-hierarchical sphere. It includes terms that represent computational systems through human social relations, institutional roles and structures of authority. This group contains expressions such as *master/worker architecture, leader/follower, primary/secondary, parent/child component, client-server, trusted agent, daemon, gatekeeper, owner, administrator, user, bot, assistant, manager, worker, broker and controller*. The source domain here is not the human body or mind, but social organisation. The target domain includes software architecture, network communication, access control, distributed systems, process management and cloud-native infrastructure.

The client-server model is among the most stable examples of social metaphor in computing. A client requests a service, while a server provides it. The terms derive from institutional and commercial relations, where a client receives assistance from a service provider. In technical discourse, this metaphor clarifies the direction of communication and the asymmetry of roles. The client initiates a request; the server responds. The metaphor is now so conventionalised that its social origin is often unnoticed, yet it continues to structure the conceptual understanding of networked interaction.

Parent/child terminology is similarly widespread in operating systems, object models, markup languages, process management and user interface hierarchies. It organises technical dependency through kinship. A parent process creates a child process; a parent node contains child nodes; a parent component passes properties to child components. This metaphor conveys origin, dependency, hierarchy and structural containment. It is usually less controversial than master/slave terminology because it describes generation and dependency rather than domination, although some inclusive language guides recommend contextual sensitivity even for parent/child usage when alternative terms would be clearer.

The terms daemon, agent and bot represent computational entities as social or semi-social actors. A daemon works in the background, often invisibly, performing persistent tasks. An agent acts on behalf of a user, system or programme. A bot performs automated actions, sometimes cooperatively and sometimes maliciously. These terms are especially important in contemporary AI and automation discourse because they

project intentionality and delegated agency onto software. The technical entity is represented not as a passive object, but as an actor with a role in a system of interaction.

Gatekeeper belongs to the social-institutional sphere. In ordinary usage, a gatekeeper controls access to a place, institution or resource. In IT discourse, the term may describe software, policy, security mechanisms or human roles that regulate access to systems, repositories, deployments, networks or data. It is particularly common in security and governance contexts. The metaphor frames access control as a social act of permission, refusal and boundary management. Trusted agent functions in a similar way. It combines the metaphor of agency with the social concept of trust, thereby representing the technical component as an authorised participant in a system.

Table 3.

Social-hierarchical metaphors in IT terminology: source models, technical domains and semantic emphasis

Microsystem	Representative IT terms	Social source model	Technical target domain	Semantic emphasis
Social-hierarchical	*client-server*	service relationship, requester and provider	network architecture, web systems, APIs	Request, response, service provision
	parent/child process, parent/child component	kinship, descent, dependency	process hierarchy, UI trees, component architecture	Origin, containment, dependency
	master/worker, leader/follower	authority, coordination, task distribution	distributed systems, clusters, replication	Control, delegation, coordination
	primary/secondary, active/passive	rank, priority, support role	databases, failover systems, replication	Priority, redundancy, continuity
	trusted agent, software agent	delegation, representation, trust	automation, AI systems, security	Authorised action on behalf of another entity
	daemon, background service	hidden servant, invisible helper	background process management	Persistence, invisibility, service
	gatekeeper, guard, policy enforcer	access control, institutional authority	security systems, identity management	Permission, refusal, boundary control
	owner, administrator, maintainer	possession, responsibility, governance	repositories, systems, services	Accountability and control

Source: created by authors on the bases of present research.

The structural-semantic taxonomy also reveals different degrees of metaphorical transparency. Some terms are strongly metaphorical and still easily recognisable as figurative, such as zombie process, orphan process, hallucination or gatekeeper. Others have undergone demetaphorisation. Terms such as memory, client, server, thread and daemon are now ordinary technical vocabulary for many specialists. Demetaphorisation occurs when repeated professional use weakens the perception of figurativeness and turns a metaphor into a conventional term. However, the loss of obvious figurativeness does not remove the underlying conceptual structure. The metaphor may no longer be consciously perceived, but it continues to organise technical reasoning.

This point is particularly important for terms such as memory and server. Most users do not experience memory as a metaphor when speaking about RAM, cache or storage. Similarly, server is now a basic technical term. Yet both terms remain historically and conceptually anthropocentric. Memory preserves the model of retention and recall; server preserves the model of service provision. Their conventionality shows that technical terminology often develops through metaphorical innovation followed by semantic stabilisation. Once stabilised, the term becomes part of the official terminological system and may be used in standards, documentation and education without any explicit reference to its metaphorical origin.

A further structural distinction may be made between nominal, verbal and adjectival anthropomorphic terms. Nominal metaphors name technical entities through human or human-related categories: virus, daemon, agent, bot, parent, child, worker, listener, orphan and zombie. Verbal metaphors describe technical operations through human-like actions: listen, call, invoke, kill, sleep, wake, crash, refuse, recover, learn and remember. Adjectival metaphors attribute qualities to systems: smart, intelligent, trusted, active, passive, blind, healthy and vulnerable. These grammatical forms differ in function. Nominal metaphors classify entities; verbal metaphors animate processes; adjectival metaphors evaluate states or capacities.

Table 4.

Structural types of anthropomorphic IT metaphors: semantic operations, discourse functions and contexts of use

Structural type	Examples	Dominant semantic operation	Function in discourse	Typical context of use
Nominal metaphors	*virus, worm, daemon, bot, agent, zombie, orphan*	Naming a technical entity through a human, biological or social category	Creates compact labels for complex entities	Cybersecurity, automation, operating systems
Verbal metaphors	*listens, sleeps, wakes, dies, crashes, invokes, refuses, learns*	Representing computation as action or behaviour	Animates invisible processes	Programming, debugging, documentation
Adjectival metaphors	*smart, intelligent, trusted, blind, healthy, vulnerable*	Assigning human-like qualities or states	Evaluates system capacity or condition	AI, security, monitoring, architecture
Relational metaphors	*parent/child, client-server, leader/follower, primary/secondary*	Structuring dependency and hierarchy	Models architecture and interaction	Distributed systems, APIs, component trees
Pathological metaphors	*infection, vulnerability, quarantine, hygiene, recovery*	Modelling system damage and protection	Explains risk and remediation	Cybersecurity and reliability engineering
Cognitive metaphors	*learning, memory, attention, inference, hallucination*	Modelling information processing as cognition	Supports explanation of AI and data systems	Machine learning, generative AI, analytics
Institutional metaphors	*administrator, owner, maintainer, gatekeeper, policy*	Representing governance and responsibility	Clarifies control and accountability	DevOps, access management, repositories
Lifecycle metaphors	*birth, creation, end of life, retirement, deprecation*	Modelling technical objects as temporal entities	Structures maintenance and replacement	Software lifecycle, data governance, product management

Source: created by authors on the bases of present research.

The problem of political correctness and demetaphorisation deserves separate attention because it shows that anthropomorphic and social metaphors are not neutral from a cultural perspective. In British and American engineering culture, there has been a visible shift towards replacing terms considered exclusionary, historically loaded or insufficiently precise. The most discussed example is the replacement of master/slave with alternatives such as primary/secondary, leader/follower, active/passive,

controller/worker or primary/replica. This tendency is not limited to informal preferences. NIST has explicitly advised avoiding terms such as master/slave because they may perpetuate unequal power relationships and has also noted that terms such as whitelist and blacklist can create clarity problems as well as bias concerns.

The IETF discussion of terminology and exclusionary language similarly treats master-slave and whitelist-blacklist as problematic examples in Internet standards discourse, arguing that more inclusive terminology can improve knowledge transfer and participation in technical communities. The matter is also visible in formal standards maintenance: RFC 9454 updates OSPF terminology in line with inclusive language practices and states that future OSPF documents should use the revised terminology. In the software industry, Git platforms and open-source communities have also moved away from master as a default branch name. GitLab, for example, announced support for main as a more inclusive default branch name for new projects, noting alignment with similar moves by GitHub and Atlassian.

The inclusive language trend should not be interpreted only as external ideological pressure on technical terminology. From a cognitive-linguistic perspective, it also reflects demetaphorisation and remapping. When master/slave is replaced by primary/secondary or leader/follower, the conceptual model changes. Master/slave foregrounds domination and subordination. Primary/secondary foregrounds rank or order. Leader/follower foregrounds coordination and direction. Controller/worker foregrounds functional distribution of tasks. Primary/replica foregrounds data replication. Each alternative selects a different cognitive frame, and therefore changes the semantic profile of the architecture. Cisco's inclusive language policy, for example, recommends replacements such as primary/secondary, primary/subordinate or control/data depending on context, which shows that the replacement is not merely lexical but architecture-sensitive.

This context-sensitive replacement is important because no single alternative can cover all technical meanings of master/slave. In replication, primary/replica may be precise. In task distribution, controller/worker or leader/follower may be better. In failover systems, active/passive may be more accurate. In device communication,

controller/peripheral may be appropriate. The replacement process therefore requires semantic analysis, not mechanical substitution. A poorly chosen inclusive alternative may reduce technical clarity. A well-chosen alternative can improve both social acceptability and conceptual precision.

Table 5.

Inclusive Terminology Replacements in IT Discourse: Semantic Shifts, Technical Contexts and Practical Comments

Earlier term	More current alternatives	Semantic shift	Suitable technical contexts	Comment
master/slave	primary/secondary	From domination to rank or priority	Databases, replication, failover	Useful when one element has priority but does not directly command the other
master/slave	leader/follower	From ownership to coordination	Distributed consensus, clustered systems	Useful when one node coordinates and others follow protocol decisions
master/slave	controller/worker	From social domination to task management	Parallel processing, job queues, orchestration	Good for architectures based on task allocation
master/slave	primary/replica	From control to data copying	Databases, storage, replication	More precise when the relation concerns replicated data
blacklist/whitelist	blocklist/allowlist	From colour-coded evaluation to explicit permission logic	Security policies, access control, filtering	Clearer and more semantically transparent
dummy value	placeholder, sample value	From potentially demeaning language to neutral function	Testing, documentation, examples	Removes unnecessary human implication
sanity check	confidence check, validity check	From mental health metaphor to procedural validation	Testing, review, verification	More precise and less ableist
grandfathered	legacy, retained, exempted	From historically loaded social metaphor to temporal status	Policy, compatibility, licensing	Clarifies whether the issue is age, exemption or compatibility

Source: created by authors on the bases of present research.

The shift towards inclusive terminology also affects the interpretation of anthropomorphism as a whole. Not all anthropomorphic metaphors are equally problematic. Terms such as parent/child, client-server, worker, agent or daemon may remain functional and broadly accepted when they provide technical clarity and do not carry strongly exclusionary associations. However, terms that encode domination, racialised colour symbolism, ableist assumptions or unnecessary violence are increasingly questioned in professional documentation. Google's developer documentation guidance recommends replacing or writing around non-inclusive terms and avoiding ableist, gendered or violent language where possible. The Inclusive Naming Initiative similarly identifies master/slave and whitelist/blacklist as highly visible problematic terms and encourages more precise alternatives.

From the standpoint of cognitive linguistics, this trend confirms that technical metaphors are not merely internal professional conveniences. They also participate in wider cultural systems of meaning. A metaphor can be technically useful, but still socially contested. Conversely, replacing a metaphor may not only remove problematic connotations but also improve conceptual accuracy. The term allowlist is more explicit than whitelist because it names the technical function directly: allowing access. Blocklist is more explicit than blacklist because it identifies the operation of blocking. In this case, inclusive terminology and technical precision support each other.

The process may be described as partial demetaphorisation. Instead of replacing one metaphor with another equally figurative expression, the terminology often moves towards more literal or functionally transparent naming. Master/slave becomes primary/secondary, controller/worker or primary/replica; blacklist/whitelist becomes blocklist/allowlist; sanity check becomes validity check. This does not mean that IT discourse is abandoning metaphor. Rather, it is becoming more selective about which metaphors are institutionally acceptable, semantically precise and professionally useful. Anthropomorphic metaphors remain deeply embedded in technical language, but they are increasingly evaluated against criteria of inclusiveness, clarity and cognitive adequacy.

The taxonomy proposed in this subchapter therefore has two dimensions. The first is semantic: anthropomorphic IT terms may be grouped according to the human-centred domains from which they draw their meanings. The second is socioterminological: terms differ in their degree of conventionalisation, acceptability and stability. Some metaphors are deeply institutionalised and unlikely to disappear, such as memory, client, server, process, thread and lifecycle. Some are productive but require careful interpretation, such as learning, attention, hallucination and intelligent agent. Others are undergoing active replacement, especially master/slave and whitelist/blacklist. This uneven development reflects the dynamic character of IT discourse.

Overall, the structural-semantic taxonomy of anthropomorphic IT terminology shows that contemporary technical language is organised by recurring metaphorical microsystems. The biological-physiological sphere models systems as living, vulnerable and recoverable entities. The psychological-intellectual sphere models computational operations as perception, memory, learning, inference and intelligence. The social-hierarchical sphere models architectures as relations of service, authority, delegation, trust and dependency. Together, these microsystems form a dense cognitive network that enables specialists to name and reason about complex digital phenomena. At the same time, the ongoing revision of contested terms shows that metaphor is not a static inheritance. It is a negotiable resource shaped by technical accuracy, professional practice and changing ethical expectations within engineering culture.

3. Pragmatic potential and functional load of anthropomorphic metaphors.

Anthropomorphic metaphors in contemporary IT discourse perform not only a nominative or cognitive function, but also a complex pragmatic role. They influence how technical knowledge is transmitted, how professional identities are constructed, how users interpret technological systems, and how developers communicate with one another in everyday problem-solving situations. In this respect, anthropomorphic metaphor should be analysed not only as a mechanism of conceptual mapping, but also as a discourse strategy that regulates the relationship between specialist knowledge and practical communication. Its functional load is especially visible in contexts where

abstract code, system architecture, artificial intelligence, cybersecurity or software behaviour must be explained to people with different levels of technical expertise.

The pragmatic value of anthropomorphic metaphor lies in its ability to transform an impersonal computational process into an intelligible event involving quasi-human actors, actions and relations. A system that "*refuses*" a connection, a script that "*does not want*" to access a database, a server that "*listens*" on a port, an agent that "*acts*" on behalf of a user, or a child class that "*inherits*" properties from a parent class are all examples of metaphorical animation. In each case, a technical entity is represented as if it possessed a limited form of subjectivity, intention or social role. This does not mean that specialists literally believe that code has consciousness. Rather, anthropomorphic language provides a practical framework for speaking about complex behaviour in a compressed, recognisable and communicatively efficient form.

The communicative function of anthropomorphic metaphors is particularly important at the boundary between professional and non-professional discourse. IT specialists often need to explain technical issues to end users, managers, students, clients or stakeholders who do not possess the same level of technical knowledge. In such situations, purely formal descriptions may be accurate but ineffective. A phrase such as "*the server did not respond to the client request within the expected timeout interval*" may be technically precise, but a simplified formulation such as "*the server stopped answering*" is more accessible. The anthropomorphic verb "*answering*" creates a bridge between the technical model of request-response communication and the everyday human model of dialogue. This bridge is not a distortion if used responsibly; it is a pragmatic adaptation to the communicative needs of the audience.

This communicative role is especially relevant in software documentation and error messaging. A good error message rarely describes the entire technical mechanism behind a failure. Instead, it often translates the failure into user-oriented language. For example, "*the application cannot find the file*", "*the system is waiting for a response*", or "*your session has expired*" are all partly anthropomorphic formulations. They assign action, expectation or temporal existence to software components. Such expressions

help the user understand what happened and what should be done next. The pragmatic goal is not full technical transparency, but actionable comprehension.

Anthropomorphic metaphor also reduces communicative asymmetry. In professional-client communication, the specialist often controls the technical code, while the non-specialist depends on explanation. Metaphorical language can reduce this imbalance by giving the non-specialist access to a familiar frame. When a cybersecurity specialist explains that a computer virus spreads through infected files, the user can immediately transfer knowledge from biological infection to digital threat. When a support engineer explains that an account has been locked because the system detected suspicious behaviour, the user can understand the situation through the metaphor of protection and controlled access. Thus, anthropomorphic terminology functions as an interpretive bridge between expert knowledge and everyday reasoning.

At the same time, the communicative function has a potential limitation. Anthropomorphic metaphors may simplify technical processes to such an extent that users develop inaccurate mental models. If a system is described as "*thinking*", "*deciding*" or "*understanding*", a non-specialist may overestimate its autonomy or intelligence. This problem is particularly acute in the discourse of generative artificial intelligence. Terms such as smart assistant, intelligent agent, machine learning, reasoning model and hallucination are useful because they make AI behaviour easier to discuss (*Kravchenko et al., 2025*). However, they may also encourage users to attribute human-like intentionality, awareness or responsibility to statistical systems. Therefore, the pragmatic effectiveness of anthropomorphic metaphor must be balanced with epistemic caution.

A useful way to represent the pragmatic load of anthropomorphic metaphor is to view it as a set of functions operating across several communicative levels.

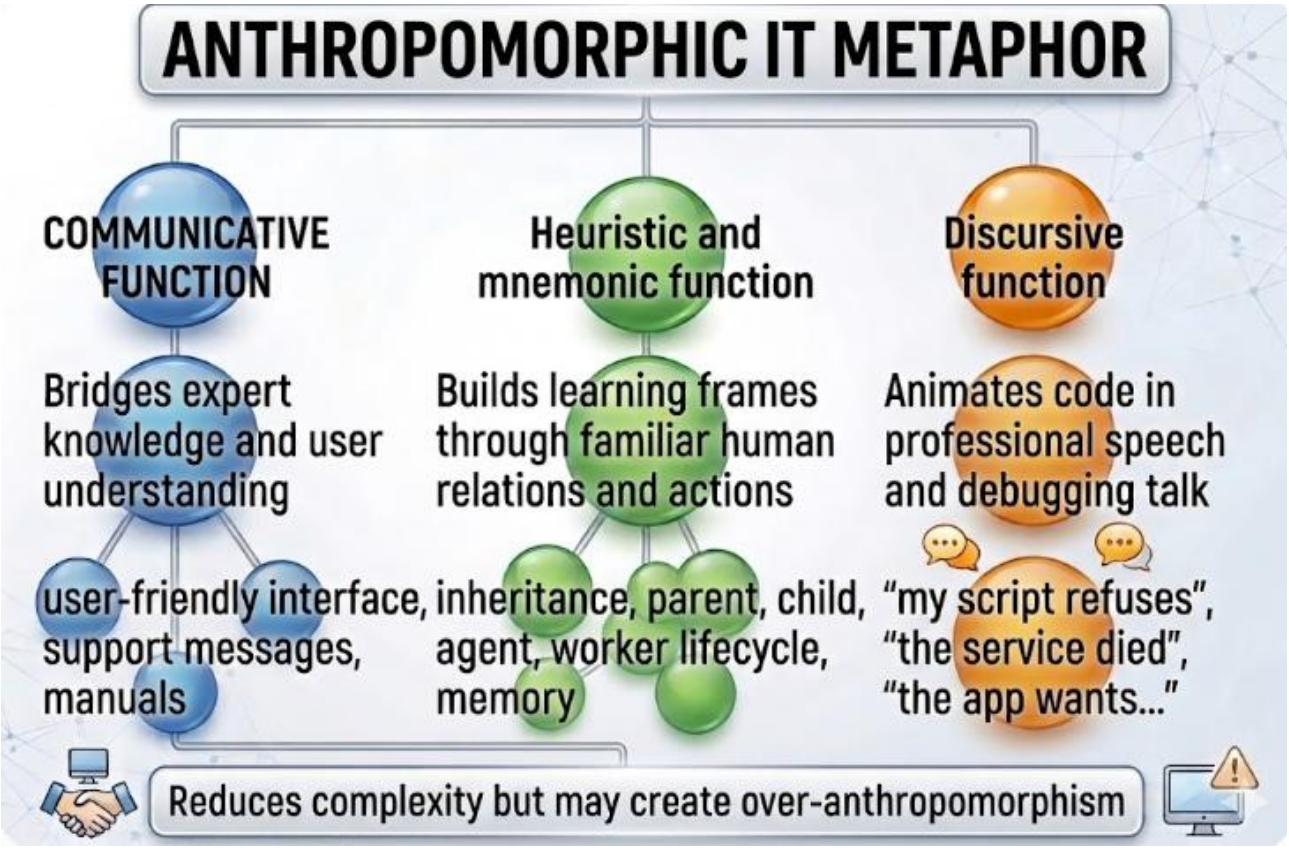

Figure 1. Functional load of anthropomorphic metaphors in IT discourse (visualisations generated using Google's Gemini AI tool).

The heuristic function of anthropomorphic metaphor is closely connected with its communicative function, but it concerns the discovery, modelling and structuring of knowledge rather than immediate explanation. In programming and software engineering, metaphors often help developers formulate hypotheses about how a system works. They guide attention towards relevant relations, possible causes of failure and potential solutions. If a process is described as a child of another process, the developer is encouraged to examine dependency, inheritance of environment variables, termination behaviour and lifecycle management. If a service is described as unhealthy, the engineer is encouraged to inspect monitoring data, resource consumption, logs, response time and availability. The metaphor does not provide the full technical explanation, but it directs inquiry.

This heuristic role is especially visible in object-oriented programming. The concept of inheritance is one of the clearest examples of a socially and biologically

grounded metaphor that structures technical understanding. In everyday human experience, inheritance involves the transfer of property, status, traits or rights from one generation or legal subject to another. In object-oriented programming, inheritance refers to a relation in which a class receives attributes or methods from another class. The terminology of parent class, child class, subclass and superclass creates a conceptual frame in which code organisation is interpreted through lineage, dependency and transmission. This frame helps learners understand that a class does not need to define all its features independently; it may obtain them from a higher-level structure.

The metaphor of inheritance performs several functions at once. First, it names a technical mechanism. Second, it creates a mental model of transfer. Third, it supports reasoning about hierarchy. Fourth, it provides a mnemonic device: learners remember the relation because it resembles a familiar social and biological pattern. The child class "inherits" from the parent class, but it may also "override" inherited methods. The verb override introduces another social-legal and behavioural metaphor: the child structure may replace or redefine inherited behaviour. This combination of inheritance and overriding makes object-oriented programming easier to conceptualise because it turns abstract code reuse into a system of relations that resembles family structure, legal succession and individual differentiation.

The mnemonic function is particularly important during onboarding. New developers entering a project must learn not only programming syntax, but also architectural conventions, codebase structure, naming practices, dependency chains and team-specific ways of speaking. Anthropomorphic metaphors provide cognitive anchors. Terms such as parent, child, worker, manager, controller, observer, listener, producer, consumer, factory, agent and mediator create familiar role-based frames. A new developer may not immediately understand the complete implementation of a distributed job queue, but the phrase "*the manager assigns work to workers*" gives an initial model of task distribution. Similarly, in event-driven programming, the term listener helps the learner understand that a component is waiting for an event and will react when that event occurs.

In this sense, anthropomorphic metaphors operate as pedagogical scaffolds. They do not replace formal instruction, but they make formal instruction more accessible. A beginner can first understand that a program "*calls*" a function, a thread "*waits*", a server "*listens*", an object "*knows*" its state, or a garbage collector "*cleans* up" unused memory. Later, these intuitive models can be refined through technical explanation. The learner moves from metaphorical understanding to operational understanding. The metaphor is therefore not an obstacle to precision if it is treated as a starting point rather than as a final definition.

The onboarding function of metaphor is also relevant in team communication. Software projects often include people with different backgrounds: senior engineers, junior developers, DevOps specialists, QA engineers, product managers, security analysts and technical writers. Each group may have its own degree of technical depth. Anthropomorphic metaphors provide shared vocabulary that allows these groups to coordinate. For instance, in a discussion about system reliability, the phrase "*the service is unhealthy*" is understandable to developers, DevOps engineers and managers. It compresses multiple possible indicators – failed health checks, high latency, excessive memory use, repeated restarts or unavailable dependencies – into one manageable expression. Further analysis may require technical detail, but the initial metaphor creates common ground.

The mnemonic function may be illustrated through the following table.

Table 6.

Anthropomorphic terms as pedagogical scaffolds in programming: human frames, technical meanings and pragmatic value in learning

Anthropomorphic term	Familiar human frame	Technical meaning	Pragmatic value in learning
parent class	family hierarchy, origin	class from which another class derives properties or methods	Helps learners understand dependency and reuse
child class	descent, inheritance	class that extends or derives from another class	Makes object-oriented relations memorable
inheritance	transfer of traits or property	reuse of methods or fields from a superclass	Connects abstract code reuse with familiar social logic
override	replacing a previous decision or rule	redefining inherited behaviour in a subclass	Helps explain controlled deviation from inherited structure
listener	attentive human receiver	component waiting for an event or input	Clarifies event-driven behaviour
worker	employee performing tasks	process or thread executing assigned jobs	Explains parallel task distribution
manager	organiser or supervisor	component coordinating resources or tasks	Supports understanding of orchestration
agent	delegated actor	software component acting on behalf of a user or system	Explains automation and representation

Source: created by authors on the bases of present research.

The pragmatic strength of these metaphors lies in their capacity to integrate multiple dimensions of meaning. A worker is not merely an executing component; the term also implies task assignment, role limitation and subordination to a coordinating mechanism. A manager implies organisation, supervision and decision-making. An agent implies delegated action. These associations can help learners build rich mental models quickly. However, they can also mislead if the metaphorical associations are taken too literally. A worker process does not possess labour rights, intention or personal responsibility. A manager component may not make independent decisions in any human sense. Therefore, technical education must both use and discipline metaphor.

The heuristic function also appears in debugging. Debugging is not only a mechanical process of locating errors. It is a form of investigative reasoning. Developers often construct narratives about what the system is doing. They may say that a service "dies after deployment", a script "cannot see the database", a process "gets stuck", a queue "starves", an application "forgets the session", or an API "refuses the request". These expressions create a provisional explanatory scenario. The developer imagines the system as an actor in trouble: it tries, fails, waits, loses, refuses or collapses. This narrative helps organise diagnostic activity. The engineer then checks logs, configurations, permissions, network routes, memory use or dependency availability.

This leads to an important discourse feature of professional oral communication: programmers often deindividualise themselves and assign subjectivity to code. Instead of saying "*I wrote a script that fails to establish a connection with the database because I probably misconfigured the credentials or network path*", a developer may say "*my script does not want to see this database*". The latter expression shifts agency from the developer to the script. Grammatically and pragmatically, the script becomes the subject of the problem. It "*does not want*", "*cannot see*", "*refuses*", "*breaks*", "*falls over*" or "*gets confused*". This personification is not merely humorous. It performs several discourse functions.

First, it reduces personal blame. In collaborative engineering environments, errors are frequent and often caused by complex interactions among code, infrastructure, dependencies and configuration. By saying "the script does not want to work", the speaker temporarily relocates agency from the human author to the technical object. This makes the problem discussable without immediate accusation. The phrase creates a socially safer space for troubleshooting. The focus shifts from "who made a mistake?" to "what is the system doing?" Such deindividualisation can support cooperative problem-solving.

Second, it reflects the opacity of complex systems. Modern software often behaves in ways that are difficult to predict from a single developer's perspective. Distributed services, asynchronous queues, container orchestration, dependency injection, caching, permissions and environment-specific configurations can create

emergent behaviour. When developers say that a system "*does not want*" to behave correctly, they are expressing not literal belief in machine intention, but the experience of dealing with a complex object whose causal structure is not immediately transparent. Anthropomorphic language becomes a way of verbalising uncertainty.

Third, the personification of code allows developers to construct rapid diagnostic narratives. Consider the phrase "*the service dies when it receives this payload*". The verb dies immediately directs attention to runtime failure, exception handling, memory limits, invalid input or dependency collapse. The phrase "*the database does not like this query*" may point to syntax errors, unsupported functions, permission restrictions, locks, timeouts or performance constraints. The phrase "*the cache forgets the user too early*" may suggest session expiry, invalidation policy or incorrect TTL configuration. In each case, anthropomorphic metaphor compresses a technical problem into a narratively manageable form.

This oral discourse is often informal, but it is not intellectually primitive. On the contrary, it reveals how specialists use metaphor to manage uncertainty, complexity and collaborative reasoning. Professional speech in software teams is full of animated code, socialised components and quasi-intentional systems. Services talk to each other, APIs complain, databases refuse, containers restart themselves, pipelines break, tests shout, builds fail, agents watch, workers pick up jobs, queues starve, and systems recover. These expressions are pragmatically efficient because they allow complex system behaviour to be discussed quickly in meetings, chats, code reviews and incident calls.

The following table shows typical examples of anthropomorphic formulations in oral professional speech.

Table 7.

Anthropomorphic expressions in informal developer discourse: technical interpretation, pragmatic function and diagnostic follow-up

Informal professional expression	Literal technical interpretation	Pragmatic function	Possible follow-up analysis
My script does not want to see the database.	The script cannot connect to the database or access it correctly.	Expresses failure without immediate blame; invites troubleshooting.	Check credentials, DNS, firewall rules, connection string, permissions.
The service died after deployment.	The service crashed, stopped or failed health checks.	Marks abrupt operational failure.	Inspect logs, resource limits, container status, dependency errors.
The API refuses this request.	The API returns an error, denial or validation failure.	Frames rejection as interactional behaviour.	Check status code, authentication, payload schema, rate limits.
The queue is starving.	Tasks are not being processed or resources are insufficient.	Highlights imbalance between workload and processing capacity.	Check workers, throughput, locks, resource allocation.
The cache forgot the session.	Session data expired, was evicted or was not stored correctly.	Makes state loss easy to discuss.	Check TTL, eviction policy, storage backend, key generation.
The build is complaining about dependencies.	The build process produces dependency-related errors.	Personifies error reporting as complaint.	Review package versions, lock files, environment settings.
The database does not like this query.	The query is invalid, inefficient or unsupported.	Softens technical failure through humorous personification.	Inspect execution plan, syntax, indexes, permissions.
The system healed itself.	Automated recovery restored service availability.	Presents resilience as bodily repair.	Check self-healing mechanisms, restart policy, failover events.

Source: created by authors on the bases of present research.

Such expressions also shape the affective dimension of professional communication. Software development is intellectually demanding and often stressful. Anthropomorphic humour helps professionals cope with system failure, time pressure and uncertainty. Saying "*the server is angry today*" or "*the test suite hates me*" introduces a playful frame that reduces tension while preserving the technical focus. This does not eliminate the seriousness of the problem, but it helps maintain group cohesion. In incident response, however, excessive informality may become

problematic if it obscures responsibility or delays precise diagnosis. Therefore, the pragmatic use of metaphor depends on communicative context.

A key distinction should be made between informal debugging talk and formal technical documentation. In oral discourse, metaphor may be expressive, humorous and approximate. In documentation, metaphor must be controlled, consistent and semantically precise. For example, a developer may say informally that "*the service gets confused*", but documentation should specify that the service enters an inconsistent state after receiving conflicting configuration values. Similarly, "*the model hallucinates*" may be acceptable as a general term in AI discourse, but research or technical documentation should define it as the generation of unsupported, false or non-grounded output. The pragmatic value of metaphor increases when its meaning is contextually disciplined.

Anthropomorphic metaphors also contribute to interface design. The idea of a user-friendly interface presupposes that systems should accommodate human cognitive limitations rather than demand that users adapt entirely to machine logic. Icons, assistants, prompts, notifications, avatars, conversational agents and help systems often rely on anthropomorphic cues. A virtual assistant may greet the user, offer help, remember preferences or apologise for an error. Even when no visual human-like figure is present, the interface may use interpersonal language: "*We could not save your changes*", "*Try again*", "*Your request is being processed*", or "*This field is required*". Such formulations create an interactional frame that makes the system appear responsive and cooperative.

However, user-interface anthropomorphism must be carefully calibrated. Too little anthropomorphism may make a system cold, opaque and difficult to interpret. Too much anthropomorphism may create false expectations of intelligence, empathy or accountability. A chatbot that says "*I understand how you feel*" may be pragmatically effective in some customer-service contexts, but it may also be ethically questionable if the system has no genuine understanding. Similarly, an AI assistant that presents suggestions with excessive confidence may lead users to overtrust its output. Thus, the

pragmatic function of anthropomorphic metaphor in interface design must be evaluated together with transparency, user autonomy and accountability.

In professional onboarding, anthropomorphic metaphor also helps transmit tacit knowledge. New developers do not learn only from manuals. They learn from the way experienced team members speak about systems. If senior engineers consistently describe a component as a “gatekeeper”, newcomers understand that its role is to control access. If a background service is described as a “worker”, they infer that it performs assigned tasks rather than coordinates the entire system. If a class is called “controller”, they expect it to regulate or direct interactions. Naming conventions therefore serve as informal architecture documentation. They encode assumptions about responsibility, dependency and expected behaviour.

This pragmatic function is especially important in large codebases. A well-chosen metaphorical name can make architecture more legible. For instance, design patterns such as Observer, Factory, Builder, Adapter, Decorator, Mediator, Proxy and Singleton are heavily metaphorical or role-based. Their names do not provide complete definitions, but they offer cognitive entry points. An Observer watches for changes; a Factory creates objects; an Adapter makes incompatible interfaces work together; a Mediator coordinates communication; a Proxy stands in for another object. These metaphors make abstract design structures memorable and teachable. They also help developers discuss architecture without repeatedly explaining implementation details.

The mnemonic role of such names may be represented as a movement from familiar social action to technical pattern recognition.

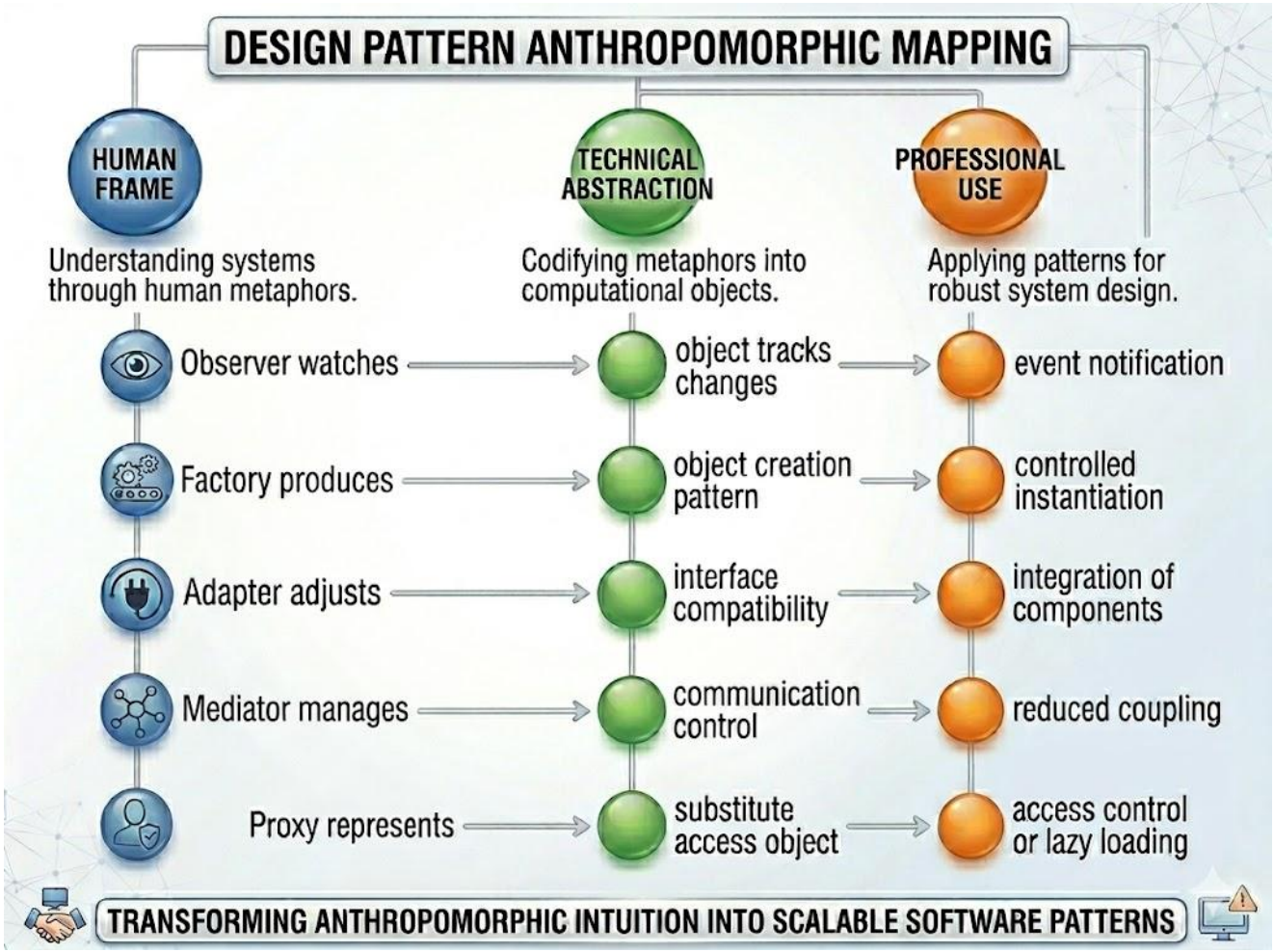

Figure 2. From human frame to technical pattern (visualisations generated using Google's Gemini AI tool).

The functional load of metaphor becomes even clearer when comparing metaphorical and non-metaphorical explanations. A purely formal explanation of inheritance might state that one class derives members from another class according to a specified type hierarchy and may redefine inherited behaviour through method overriding. This formulation is precise, but cognitively dense for beginners. A metaphorical explanation says that a child class inherits features from a parent class and can change some of them. This is less formal, but more accessible. The best pedagogical practice combines both: the metaphor introduces the structure, while the formal explanation defines its technical limits.

In team discourse, anthropomorphic metaphors also support distributed cognition. Software development is rarely an individual mental activity. It is distributed

across people, tools, repositories, issue trackers, documentation, automated tests, CI/CD pipelines and monitoring systems. Anthropomorphic language helps coordinate this distributed environment by assigning roles to non-human participants. The pipeline "*checks*" the code, the test "*catches*" regressions, the linter "*complains*", the monitor "*watches*" the service, the alert "*tells*" the team that something is wrong, and the bot "*opens*" a pull request. These expressions integrate tools into the communicative ecology of the team. They become quasi-participants in the workflow.

This has practical consequences for responsibility and agency. In DevOps culture, automated systems are designed to perform actions that previously required human intervention. Deployment bots, monitoring agents, auto-scalers, self-healing systems and security scanners act within predefined boundaries. Anthropomorphic terms reflect this delegation of operational agency. A bot can merge a dependency update; an agent can collect telemetry; an orchestrator can restart containers; a watchdog can detect failure. The language of agency corresponds to a real shift in technical practice: software components increasingly perform tasks that resemble human operational roles, even though they do so without consciousness or intention.

The discourse of artificial intelligence intensifies this tendency. When AI systems are described as assistants, copilots, agents, tutors or advisors, they are positioned as collaborators rather than mere tools. This positioning has strong pragmatic effects. Users may engage with them more naturally, ask questions, accept suggestions and treat their output as communicative response. In educational or professional contexts, this may support productivity and learning. At the same time, it may blur responsibility. If an AI "*suggests*" faulty code, who is responsible for the error: the user, the developer of the system, the organisation deploying it, or the model itself? Anthropomorphic language tends to distribute agency in ways that require (*Stepanova et al., 2025*) critical examination.

The pragmatic potential of anthropomorphic metaphor is therefore ambivalent. On the positive side, it enhances accessibility, supports learning, improves memorability, facilitates teamwork, reduces cognitive load and helps users interact with complex systems. On the negative side, it can oversimplify causality, obscure

technical mechanisms, shift blame away from human actors, encourage overtrust in automation or create misleading impressions of machine intelligence. This ambivalence does not make anthropomorphic metaphor undesirable. Rather, it shows that such metaphor must be analysed functionally and contextually.

The following table summarises the major pragmatic functions and their possible risks.

Table 8.

Pragmatic functions of anthropomorphic metaphors in IT discourse: mechanisms, positive effects and potential risks

Function	Mechanism	Example	Positive effect	Potential risk
Communicative	Translates technical behaviour into human action frames	The server *answers* the request.	Makes system behaviour accessible to users.	May hide technical complexity.
Heuristic	Guides diagnostic reasoning through familiar scenarios	The service *died* after deployment.	Helps form hypotheses about failure.	May encourage vague explanation.
Mnemonic	Links abstract concepts with familiar relations	A child class *inherits* from a parent class.	Supports learning and onboarding.	May create over-literal understanding.
Social	Reduces blame and supports team problem-solving	My script does not *want* to connect.	Makes errors discussable without personal accusation.	May obscure human responsibility.
Interface-oriented	Frames systems as cooperative partners	User-friendly interface, assistant, *help* bot.	Improves usability and user engagement.	May create excessive trust.
Organisational	Assigns roles to components in architecture	worker, manager, controller, gatekeeper.	Clarifies responsibility and dependency.	May encode contested social hierarchies.
Affective	Adds humour and reduces tension	The database is *angry* today.	Supports group cohesion under stress.	May be inappropriate in formal reporting.
Epistemic	Models AI behaviour through cognition	The model *hallucinates*.	Provides a recognisable label for false output.	May anthropomorphise machine output too strongly.

Source: created by authors on the bases of present research.

In oral professional speech, the phrase "my script does not want to see this database" deserves special attention because it brings together several pragmatic mechanisms. The script is personified as an unwilling subject. The database is represented as an object of perception, something that can be "seen" or not seen. The developer is partially removed from direct responsibility. The technical problem is framed as a failed relationship between two entities. This expression is semantically inaccurate if read literally, but highly effective in context. It quickly communicates that there is a connection or access problem, probably involving configuration, credentials, network visibility or permissions. It also invites collaborative troubleshooting: colleagues can ask whether the script has the right connection string, whether the database is reachable from the environment, whether firewall rules allow access, or whether credentials are valid.

Such phrases show that anthropomorphic metaphor often works through dramatization. Code becomes an actor, the database becomes an interlocutor or object of perception, the system becomes an environment, and the developer becomes a mediator observing the conflict. This dramatization is pragmatically useful because it turns abstract failure into a scene. Human cognition is well adapted to scenes involving actors, actions, obstacles and goals. Technical systems are therefore narrated as if they were small dramas: *the client asks, the server refuses, the worker waits, the queue starves, the monitor complains, the process dies, the watchdog restarts it, and the system recovers*. These narratives make complexity manageable. The same pattern appears in incident communication. During outages, teams often construct a timeline in which services and components become actors: "*the API started returning errors*", "*the queue stopped draining*", "*the workers could not keep up*", "*the database began rejecting connections*", "*the failover mechanism kicked in*", "*the system recovered after restart*". This language is technical enough for professionals but still narrative enough to support coordination under pressure. It allows participants to reconstruct causality and sequence. Later, in a formal post-incident report, the same events may be rewritten in more precise terms. Nevertheless, the initial anthropomorphic narrative often plays a crucial role in real-time sense-making.

From a cognitive-pragmatic perspective, anthropomorphic metaphors also support compression. Technical explanations can be long, but professional interaction often requires speed. In a stand-up meeting, code review or incident call, participants need short expressions that carry dense meaning. "*The worker is stuck*" may compress information about blocked execution, unavailable resources, deadlock, queue backlog or dependency failure. "*The model forgot the context*" may compress information about context-window limits, retrieval failure or state loss. These expressions are not complete diagnoses, but they are efficient signals that direct attention.

The functional load of anthropomorphic metaphor also differs by genre. In textbooks and tutorials, metaphor supports learning. In documentation, it supports clarity but must remain controlled. In code comments, it may clarify intent. In variable or class names, it encodes architectural roles. In oral debugging, it supports rapid reasoning and social negotiation. In user interfaces, it shapes interactional expectations. In marketing discourse, however, anthropomorphic metaphor may become more problematic because it can exaggerate capability. Words such as intelligent, autonomous, human-like, understands and thinks are often used persuasively. In such contexts, the boundary between explanatory metaphor and promotional overstatement becomes fragile.

For this reason, responsible technical communication should distinguish between pragmatic metaphor and ontological claim. Saying that a server "*listens*" is a pragmatic metaphor; it does not imply auditory perception. Saying that an AI system "*understands*" a legal document may be interpreted as a stronger claim about cognitive capacity. The former is conventional and relatively safe; the latter requires qualification. A cognitive-linguistic approach helps identify these differences by asking which features are mapped from the human domain to the technological domain, which features are excluded, and how the audience is likely to interpret the expression.

The pragmatic potential of anthropomorphic metaphor is therefore inseparable from audience design. When speaking to expert developers, metaphor may be informal, compressed and technically saturated. When speaking to end users, it must be accessible but not misleading. When writing academic analysis, it must be explicitly defined. When

designing interfaces, it must support usability without creating false expectations. When onboarding junior developers, it must provide cognitive scaffolding while gradually introducing formal precision. The same metaphor may be helpful in one context and problematic in another.

In all, anthropomorphic metaphors in IT discourse carry a substantial functional load. They serve as communicative bridges between expert code and user understanding, as heuristic tools for technical reasoning, as mnemonic devices for learning and onboarding, and as discourse strategies in oral professional communication. They help developers and users speak about invisible computational processes as if they were structured, acting and interacting entities. They also enable teams to manage complexity, uncertainty and affective pressure in collaborative work. At the same time, their use requires critical awareness. Anthropomorphic metaphor is most valuable when it supports comprehension without replacing technical explanation, when it reduces cognitive load without producing false beliefs, and when it animates code without obscuring human responsibility. In this balanced form, it remains one of the most powerful pragmatic resources of contemporary IT discourse.

Conclusion. The analysis undertaken in this work has shown that anthropomorphic metaphorisation in contemporary IT discourse is not a chaotic or marginal linguistic phenomenon. On the contrary, it functions as a structurally organised cognitive system through which abstract computational entities are conceptualised, named and explained. The examined terminology demonstrates that anthropomorphic metaphors are concentrated around several dominant frames, among which the social-hierarchical and biological-physiological models are especially productive. Terms such as parent process, child component, client-server, trusted agent, daemon, worker, computer virus, zombie process, orphan process, kill a process, hygiene and data lifecycle reveal that the IT terminological system actively draws upon human relations, social roles, bodily vulnerability, life, illness, death and recovery. These metaphors do not merely embellish technical discourse; they provide stable conceptual patterns for organising knowledge about complex digital systems.

The findings confirm that metaphor is one of the principal cognitive tools for reducing the level of abstraction in IT discourse. Computational systems, software

processes, data structures, networks, artificial intelligence models and cybersecurity mechanisms are often invisible, distributed and difficult to perceive directly. Anthropomorphic metaphor makes them cognitively manageable by converting abstract operations into familiar scenes of action, interaction, dependency, hierarchy, perception, failure and restoration. A server may "*listen*", a process may "*sleep*", a model may "*learn*", a system may "*recover*", and a component may "*inherit*" properties from another component. Such expressions are metaphorical in origin, yet they become functionally precise within professional communication (*Nykyporets et al., 2025*) because they help specialists reason about technical behaviour quickly and economically.

The theoretical significance of the study lies in extending conceptual metaphor theory and conceptual integration theory to the material of contemporary computer linguistics and terminology of the 2020s. The chapter demonstrates that IT discourse is a highly productive field for cognitive-linguistic analysis because it combines formal precision with metaphorical creativity. The donor domain HUMAN BEING is projected onto the target domain COMPUTER TECHNOLOGY through a set of recurrent mappings: body → system structure, health → security and reliability, family → process or component hierarchy, agency → software behaviour, cognition → artificial intelligence, and social role → architectural function. These mappings show that technical terminology is not detached from human embodied and social experience. Rather, it is formed through the systematic adaptation of such experience to new technological realities.

The practical value of the study is particularly relevant for technical writers, translators and specialists involved in the localisation of IT products. Anthropomorphic terms should not be translated mechanically, because their cognitive and pragmatic functions depend on context, genre and target audience. In user documentation, such metaphors may increase clarity and make interfaces more accessible. In formal standards, security documentation or AI-related texts, however, they require careful control in order to avoid ambiguity, exaggeration or misleading personification. Translators should therefore consider not only the lexical equivalent of a term, but also the conceptual frame it activates in the target language. For example, terms such as agent, daemon, worker, parent, child,

memory leak, hallucination and smart system may require different translation strategies depending on whether the context is technical documentation, academic analysis, interface text or marketing communication.

The results are also important for the creation of new terminology in artificial intelligence and cybersecurity. These domains are developing rapidly and constantly produce new phenomena that require naming. Since metaphor strongly influences how users and professionals understand technological systems, term formation should take into account not only brevity and technical accuracy, but also cognitive adequacy. In AI discourse, especially, anthropomorphic terms such as learning, reasoning, memory, attention, hallucination, agent and assistant may be useful, but they can also encourage over-attribution of human-like cognition to computational systems. In cybersecurity, biological metaphors such as infection, hygiene, immunity, quarantine and recovery remain powerful explanatory tools, yet they must be used with precision to avoid oversimplifying complex threat models. Thus, responsible term creation requires a balance between accessibility, conceptual clarity and epistemic caution.

Further research may develop this analysis in several directions. One promising perspective concerns the influence of generative models and large language models on the direction of metaphorisation in technical discourse. Traditionally, machines have been described through human metaphors: they learn, listen, remember, decide, hallucinate or assist. However, the growing cultural presence of AI may produce the reverse tendency, in which human cognition, communication and labour are increasingly described through computational metaphors. This reverse anthropomorphism, or technomorphic conceptualisation of the human, may become especially visible in expressions that describe people as processors of information, users of prompts, generators of output, optimisable agents or systems with limited bandwidth. Future studies should therefore examine not only how humans linguistically humanise machines, but also how machine-centred models begin to reshape the language used to describe human intelligence, creativity and social interaction.

References:

Fauconnier G., Turner M. The Way We Think: Conceptual Blending and the Mind's Hidden Complexities. New York: Basic Books, 2002. URL: https://openlibrary.org/books/OL7594119M/.

Geeraerts D., Cuyckens H. (eds.). The Oxford Handbook of Cognitive Linguistics. Oxford University Press, 2007. DOI: https://doi.org/10.1093/oxfordhb/9780199738632.001.0001.

Ibrahimova L. V., Nykyporets, S. S. Information security in the global context: linguistic perspectives and the role of English. *International security studios: managerial, technical, legal, environmental, informative and psychological aspects.* International collective monograph. Volume II. ISAP, Research and Education. 2025. 436 p., P. 321-345. DOI: https://doi.org/10.5281/zenodo.15356365.

Kot S. O., Nykyporets, S. S. Activating students' cognitive engagement in technical English learning with AI tools. *Science and education in the third millennium: information technology, education, law, psychology, social security and work, management. International collective monograph.* Volume I. Institute of Public Administration Affairs. Lublin, Polska, 2025. 532 p., Pp. 295-332. DOI: https://doi.org/10.5281/zenodo.16942267.

Kravchenko K., Ketsyk-Zinchenko U., Suduk I., Nykyporets S., Cherednychenko V. Effectiveness of online platforms in developing language skills of higher education students. *Revista Eduweb*. 2025. 19(3). P. 303-314. DOI: https://doi.org/10.46502/issn.1856-7576/2025.19.03.19.

Lakoff G., Johnson M. Metaphors We Live By. Chicago: University of Chicago Press, 2008. 308 p. URL: https://surl.li/vrdxfq

Lakoff G., Johnson M. The metaphorical structure of the human conceptual system. *Cognitive Science*. 1980. Vol. 4, no. 2. P. 195-208. DOI: https://doi.org/10.1207/s15516709cog0402_4.

Nykyporets S. S., Stepanova I. S., Hadaichuk N. M., Derun V. H., Medvedieva S. O. Evolving expressions: contemporary trends in English phraseology and their sociocultural implications. *Bulletin of Science and Education. Series «Philology».* 2024. № 2(20). Pp. 19-33. DOI: https://doi.org/10.52058/2786-6165-2024-2(20)-19-33.

Nykyporets, S. S., Kukharchuk H. V. Intercultural communication in information security: risks, conflicts, and educational opportunities for English language teachers. *International security studios: managerial, technical, legal, environmental, informative and psychological aspects*. International collective monograph. Volume II. ISAP, Research and Education. 2025. 436 p., P. 398-420. DOI: https://doi.org/10.5281/zenodo.15356424.

Sachaniuk-Kavets'ka N. V., Nykyporets S. S. LLM-based automation for translating mathematical formulae and symbols: challenges and perspectives for technical communication. *Scientific innovations and advanced technologies. Series «Education/Pedagogy»*, 2026. № 3(55). P. 660-677. DOI: https://doi.org/10.52058/2786-5274-2026-3(55)-660-677.

Stepanova I. S., Nykyporets S. S., Hadaichuk N. M. Exploring the evolving dynamics of axiological concepts in the modern linguistic space: a comprehensive scientific analysis. *Modern Ukrainian linguospace: ethnomental, axiological, pragmatic aspects: Scientific monograph*. Riga, Latvia : Baltija Publishing, 2023. P. 162-190. DOI: https://doi.org/10.30525/978-9934-26-365-1-9.

Stepanova I. S., Nykyporets S. S., Hadaichuk N. M., Boiko Y. V., Slobodianiuk A. A. Hedging and epistemic modality in academic discourse: linguistic markers of caution and non-categorical claiming. *Bulletin of Science and Education. Series "Philology".* 2025. №12(42). Pp. 140-155. DOI: https://doi.org/10.52058/2786-6165-2025-12(42)-140-155.

Stepanova I. S., Nykyporets S. S., Kukharchuk H. V. Integrating artificial intelligence tools into project-based English language instruction for technical students: a framework for fostering critical and creative thinking. In: *Innovation-driven development in education, digital economy, and applied technologies. Monograph.* Editors: Aleksander Ostenda, Dominika Kalita. The University of Technology in Katowice Press, 2025. P. 208-215. DOI: https://doi.org/10.54264/M055.