acrt AI, Computer Science and Robotics Technology

IntechOpen Journals

MINI-REVIEW

Artificial Intelligence Theories: Application to CommonKAD Methodology

Wangai Njoroge Mambo*

United States International University Africa, Nairobi, Kenya
*Correspondence: E-mail: mambown@protonmail.com

Citation
Wangai Njoroge Mambo (2024), Artificial Intelligence Theories: Application to CommonKAD Methodology. *AI, Computer Science and Robotics Technology* 3(1), 1–18.

DOI
https://doi.org/10.5772/acrt.2022035

Received: 25 March 2024
Accepted: 3 June 2024
Published: 12 September 2024

Abstract

Theories are required for artificial intelligence (AI) to make greater progress. Despite the development of several AI theories, their use is minimal and their nature is not widely known. An analogy with software engineering theories was used to analyze kernel, genetic, design decision, task, and AI innovation theories to determine their nature and characteristics. These theories were then applied to the CommonKAD methodology in AI to explore how they could improve the methodology, potentially contributing to the evolution of AI theories and increasing their application.

Keywords: AI intelligent systems, AI philosophy, CommonKAD, design theories, nature of AI theories

1. Introduction

Theories are a language for comprehending the world; they shape what we observe and perceive. A good theory is an asset while a wrong one can be a liability [1]. A theory is a system of rules that mimic the real world in a cost-effective and painless manner [2]. They enable sense-making, organizing, and leveraging large amounts of knowledge. According to the postulates of systemism, all things or ideas are put together into theories and artifacts that function as systems [3]. Ideas organized as systems (theories) are more effective and efficient in achieving their goals than those operating in isolation.

Artificial intelligence (AI) is a science and engineering discipline [4]. The AI, software engineering (SE), and science and engineering disciplines require theories to advance as disciplines [5]. In the early days, civil, electronics, and aeronautical engineering lacked explicit theories and relied on rules of thumb and trial and error, making their work unpredictable. These fields are being developed rapidly by creating theories with explanatory and predictive power over phenomena [2]. AI and SE are similar in that they both involve science and engineering, but AI has a nearly equal balance of both, whereas SE is predominantly engineering-focused. Hence, AI should ideally have more science theories or hybrids of science and engineering compared to SE. Also, there are more similarities shared between SE and AI compared to other computing disciplines such as computer science, computer engineering, information systems (IS), and information technology.

AM/Eurisko is a program based on the principle that research is learning, introducing a discovery-based learning approach [6]. Scientists form research communities that create, apply, evolve, and retire theories that drive the growth of their disciplines. The researchers Charniak and McDermott [6] applied this principle to AI theory building, suggesting that research in AI is analogous to the work of scientists.

This indicates that scientists learn through research to create theories. The authors used the principle to determine implications for AI theory building. They considered three philosophical theory change perspectives that apply to AI theory:

(1) **Popper's Theory of Falsification**: Theories are tested and falsified through experiments. Falsified theories are discarded and replaced with new ones. This implies that while theories can be proven wrong, they can never be proven right [7].
(2) **Kuhn's Paradigm Shift**: New fields develop new theories through paradigm shifts, creating completely new frameworks for understanding [8].
(3) **Lakatos' Research Programs**: A theory is replaced when a better theory emerges, suggesting that experiments do not play a core role in theory change [9].

The derivation and application of these principles demonstrate a concurrent analogy between AI solutions and human reasoning. While human reasoning has predominantly inspired the development of AI solutions, the reverse derivation of principles is largely an unexplored area. Analogy-inspired discovery, invention, and innovation are widely used in science, engineering, and technology to solve new problems based on existing solutions.

Artificial intelligence's long road to becoming a more sustainable and mature discipline involves creating new theories and evolving existing ones. This journey may seemingly have a long path as AI deals with more complex intelligence

phenomena than fields like physics or electronics and AI phenomena are not governed by natural laws except in hardware domains like robotics. Physics researchers have made great progress, but AI is unlikely to make similar advances quickly due to the complexity of many causes and variables than physics that have emerged from heterogeneous processes. Thus, AI-theory-building progress will be slow and based on inventing, combining, and reorganizing an ever-increasing collection of incomplete theories [10]. These incomplete theories will evolve into micro-, mid-range, and kernel (general) theories. Although incomplete theories can be useful for further development, these theories are not as effective as complete theories, and combining them is less fruitful than working within a system of completed theories. The perspectives of Charniak and McDermott [6], as well as Minsky [10], on theories apply to varying degrees to AI engineering and science paradigms. The theories analyzed in this study encompass science, engineering, design, and hybrid categories. Most engineering and design theories cannot be falsified due to the broad range of variables and causes. However, their productivity may be demonstrated through widespread use among researchers and practitioners.

A fourth philosophical perspective, distinct from falsification, paradigm shifts, and the creation of better theories, suggests that AI theories develop through dialectics, in which a set of theories is proposed as the thesis, opposing theories form the antithesis, and the synthesis stage combines the thesis and antithesis, harmonizing them. In this synthesis, unproductive parts of theories are removed and contradictory yet useful elements are harmonized. In the work proposed by Johnson [11], five paradigms of computing have been discussed: the dialectical thesis, antithesis, and synthesis; proof by demonstration (where building something demonstrates its viability); empiricism (develops theory by generating hypotheses and validating them with data); hermeneutics (involves creating and operating artifacts in real environments); and mathematical proof (involves creating and proving theories mathematically). Theories of change, falsification, paradigm shift, and the creation of better theories are broadly applicable across different disciplines. Meanwhile, Johnson's [11] dialectics, hermeneutics, and empiricism are general approaches adapted from outside computing, while mathematical proof and proof by demonstration are specific to computing.

The SE shares many similarities with AI as both involve developing software that executes on computers and acquiring knowledge and experience by learning from doing, inventing, and then systemizing what is learned. The limited use of theories in both disciplines has been identified as an obstacle preventing their maturation as engineering disciplines [5, 12]. Finally, disciplines and other fields should impart knowledge to one another.

The Institute of Electrical and Electronics Engineers (IEEE) [13] defines SE as a systematic, disciplined, and quantifiable approach to the development, operation,

and maintenance of software, emphasizing the application of engineering principles to software. By substituting "SE" with "AI" and adapting "software" to "intelligent software" and "engineering discipline" to include both science and engineering, the derived definition would qualify as an AI definition.

Since its inception, SE has aspired to be recognized as an engineering discipline while AI has aimed to replicate human intelligence in machines, integrating both science and engineering. Early definitions of AI were focused on the technology of creating machines that think and act like humans, imitating complex human skills, or making intelligent machines. Others defined AI by analogy to human intelligence, deep learning, machine learning, and cognition. The author McCarthy defined AI as the science and engineering of making intelligent machines [14]. Initially, AI emphasized scientific exploration more than engineering, contrasting with SE's early focus on engineering principles with less emphasis on science.

CommonKAD is an expert, knowledge-based, [15] and knowledge management [16] systems development methodology created by the European ESPRIT project. It contrasts with agile methods by emphasizing comprehensive planning and designing. As a comprehensive literature review of AI methodologies, the AI theories discussed in this study will also cover several aspects of CommonKAD. This study selected CommonKAD to apply AI theories, aiming to understand the nature of these theories and discover ways to improve them, while reporting the advantages and disadvantages of the aforementioned CommonKAD methodology. AI can benefit from learning through analogy with SE by creating relationships between AI theories and CommonKAD similar to those between agile methods and Software Engineering Method and Theory (SEMAT) SE theory. The SEMAT SE theory enhances agility by, for example, providing teams with tools to improve their working methods by comparing and contrasting practices [17]. The study's exploration and application of AI theories to improve CommonKAD is partially inspired by the way SEMAT theory is used to improve agile methods.

Theories analyzed in this study include kernel, genetic, design decision, task, and innovation theories. Kernel theories encompass most intelligence activities; genetic, decision design, and innovation theories are domain-specific while task theories are domain-independent and address the basic, smallest atomic unit of an activity. Each theory will be used to explore possible ways of understanding, discovering, and improving CommonKAD. Methodologies are constructed by combining microtheories, best practices, and design advice while theories are built by integrating concepts and theory fragments. The study aims to: (1) determine the nature of AI theories; (2) explore how analogy can be used to improve AI theorizing by learning from engineering and software engineering theory building; and (3) investigate how these theories can contribute to understanding and discovering

possible ways to improve the CommonKAD methodology, with the possibility of applying lessons learned to improve the theories.

2. Designing and creating theories

This section addresses the first and second research objectives by analogizing the nature of theorizing and learning with engineering and SE theory building.

2.1. AI philosophy and relationship with design theory

Philosophies such as those proposed by Popper, Kuhn, and Lakatos apply broadly to most disciplines. In contrast, specific philosophies apply to disciplines like AI or related fields like computing and engineering. The AI philosophy, traditionally framed within the philosophy of science, should be expanded to include engineering and technology, contributing to the philosophy of engineering [18]. While transdisciplinarity is based on duality, [19] AI involves both duality and either–or logic. Design theory, central to the philosophy of technology [20], should be integrated into AI philosophy to encompass engineering aspects. Design theory, initially developed within design disciplines, has been adopted by fields such as AI as both science and engineering theory. Design science research frameworks contribute to creating new knowledge through scientific research methods and also innovative artifacts through engineering methods [21]. These design research methods are used in all computing disciplines [22], often being adapted or developed based on existing computing theories. Some of these are frameworks that, in addition to research, create artifact innovations. When theories and method-based theories are applied across computing disciplines, the underlying concepts are implicitly applied. Computing disciplines sometimes borrow methods from other disciplines and later adapt these methods to fit computing theories [22]. This ability to create theories and methods applicable to all computing disciplines shows that these disciplines have many similarities. This interconnectedness can be used by bioinspired design to create theories for one computing discipline from the existing theories of other computing disciplines.

Wieringa *et al.* [23] created the first SE design theory with inspiration from engineering theories, which are practical and widely usable as design theories. Similarly, AI design theories could be adapted to function as design research and innovation theories as demonstrated with IS design research [24]. This adaptation could streamline AI artifact research, development, and innovation.

The integration of philosophy and AI theory has become mainstream [25], with AI philosophy essential for neuroscience and neurophysiology to successfully reverse-engineer human and mammalian brains and to advance robotics [26]. Reverse engineering neuroscience and neurophysiology will benefit AI and other

disciplines, especially through inspired design. AI theory is crucial for systematically transforming basic research into applied research and inventions, complementing AI philosophy.

The classical "computationalist" view is that AI and cognitive science are two sides of the same coin, with computing representations applicable across computing natural, artificial, and cognitive science systems [25], reflecting a transdisciplinary duality. Various paradigms, such as connectionist and symbolic, can design intelligent agents [27]. These paradigms may eventually be replaced by others such as behavioral robotics and neural networks [25]. Theories should ideally be derived from philosophy, and designers use design strategy to translate user and AI knowledge into AI products [28], leveraging multiple perspectives and alternatives.

Engineering design theories can be considered as user-friendly as they elucidate phenomena in terms of mechanisms, while most SE theories are statistical theories that are difficult to use [23]. The theories analyzed in this study are categorized as incomplete design theories to varying degrees. Previously, the author Wang [29] suggested that there are no widely used design theories in AI. In the aforementioned work, one of the potential reasons for this is that design theories are more difficult to use than design methods as design theories are formulated at a higher abstraction level than design methods. Another reason is that creating computing design theories is a more recent endeavor compared to the longstanding use of design methods. Currently, only pioneering innovators are experimenting with and adopting these theories, often through a process of trial and error and experimentation [30]. The lack of a critical mass of early adopters makes it difficult for subsequent groups of adopters as these groups rely on the knowledge and experience of early adopters.

Design theories have four characteristics: they address design issues, are generative, accept the propositions and language of other design theories, and utilize propositional logic [31]. These theories create solutions and designs, making them observable and understandable to adopters. AI design theory could incorporate elements from other design theories, building on existing research. The transfield between AI and SE could provide a basis for these disciplines to use each other's theories [12].

2.2. How are AI and SE theories created?

Mature scientific disciplines build and accumulate knowledge primarily through theory development. In contrast, SE has historically placed less emphasis on theory building than these disciplines, and AI even less so [32]. For example, the annual SEMAT workshop focuses on general theory development in SE, starting with identifying the SE kernel common to all software engineering [33]. An equivalent

focus on theory development in AI through workshops, conferences, or journals could play a similar role.

Science, engineering, and computing follow the same Technology Innovation, Development, and Theory Creation (TIDTC) cycle [5], which makes it possible for AI to learn from SE theory building. For example, the SEMAT SE kernel theory prevents teams from being constrained by methods by continuously improving them [33]. AI theories could similarly enhance the CommonKAD methodology. While there has been some borrowing between SE and AI, there is undoubtedly room for improvement. Borrowing theories is widely recognized as beneficial, though opinions differ on which and how many theories to borrow. The AI and SE communities should engage more through collaboration and share knowledge during major development and research projects across the TIDTC cycle steps: folklore, knowledge codification, and theory development.

The Function–Behavior–Structure (FBS) engineering design theory, initially from engineering [34], has been borrowed by SE. Although there are few studies on theory borrowing in computing, IS have conducted several investigations. For example, studies have found that even inappropriate theory borrowing may be useful as a learning process, developing theory-borrowing skills [35]. The authors Hall and Rapanotti [36] have found that borrowing similar ideas and knowledge from IS by SE is useful, and similarities between computing disciplines can be explored for potential borrowing. SE's rational unified process, Pahl engineering, and service design models when cast into FBS revealed many similarities, indicating that design may be independent of the discipline [37]. This disciplinary independence is a transdisciplinary principle. The similarities between design theories and some design elements across disciplines suggest opportunities for borrowing and extending some computing design theories. In the study by Ralph [34], combining SE sense-making with FBS design theories for modeling software systems has been proposed. The FBS theory proponents claim that the theory is based on eight fundamental design processes [34, 38]. In addition, FBS has been applied in different disciplines, providing some evidence supporting this claim.

Theories make knowledge more transferable [39], and are important for conceptualization and communication within a research field, aiding practitioners in making strategic technology and project decisions [40]. The SE theories have a higher degree of diversity than those of physics [39], but transdisciplinarity can help reduce this variation [12]. AI is transdisciplinary [41], with a broader range of theories compared to SE. The goal of intelligent SE is to apply SE knowledge to AI and vice versa [42], increasing opportunities for theory use in both disciplines. Figure 1 shows the evolution of theories and methods.

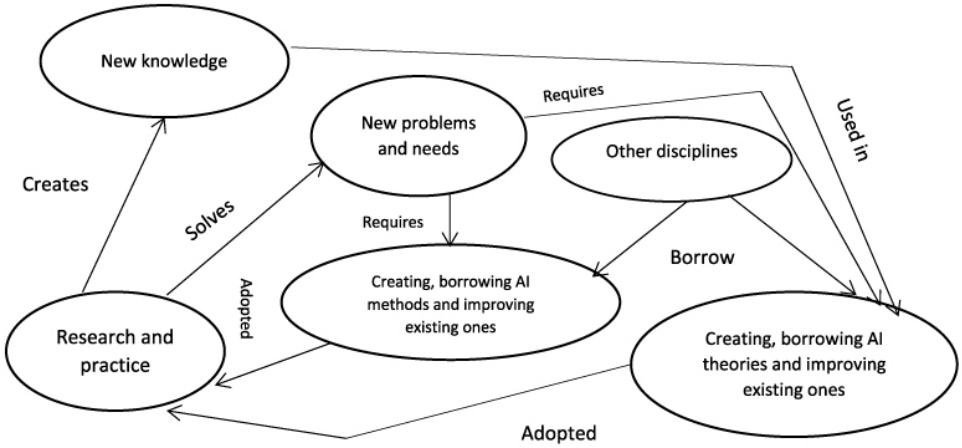

Figure 1. Theory and method creation, adoption, and evolution.

Furthermore, the author Gruner [43] discussed that general theories of SE and IS design could not exist, and any attempt to develop such theories is ineffective and wastes resources. Also, the authors Shaw and Garlan [5] discussed developing SE to mature the discipline into a true engineering field. Furthermore, in the study by Gruner [43], the development of microtheories has been supported. In contrast, Shaw and Garlan [5] advocate for three types of theories: general, mid-range, and microtheories. There is a contradiction between these two studies on general theories and agreement on microtheories. Moreover, Gruner [43] does not consider mid-range theories. The contradiction can be resolved with SE dialectics. In addition, Shaw and Garlan [5] present the thesis for general theories while Gruner presents the antithesis. Another study supporting the thesis for the development of general theories proposed a unified theory of intelligence, unifying artificial, human, and other types of intelligence [44]. A synthesis could resolve the contradiction by, for example, stating situations when developing general theories is appropriate and when it is not.

Premature theorizing is likely to be wrong but not fruitless and is preferable to delayed theorizing, which can result in more significant issues [32]. Failure and success in theorizing produces useful lessons and helps develop theory-building skills. The same applies to borrowing the wrong theories [35]. Failure helps discover what does not work while delayed theorizing prolongs the creation of knowledge on shaky foundations, leading to diminishing returns on research efforts.

Furthermore, Shaw and Garlan [5] developed their general science–engineering cycle theory method by using analogies from engineering. General theories of theory building apply across science, engineering, and technology disciplines and, therefore, to AI. Some non-general engineering and software engineering theories can be generalized or unified to make them applicable to AI.

3. Analysis of selected AI theories

This section addresses the third research objective by analyzing five types of AI theories and exploring how they can assist in understanding and improving CommonKAD.

3.1. Kernel artificial intelligence theories

The kernel theories are general and cover a broad range of phenomena. Three kernel AI theories are analyzed here.

Intelligent decision-making theory (IDMT) unifies decision-making theory with known probability distribution and Solomonoff's universal induction theory with unknown probability distribution [45]. Unification enables the theory to deal with problems with known and unknown probability distributions. The IDMT is agent-oriented and based on a universal agent that learns by inferring and acting using decision theory and induction.

Unified intelligence theory (UIT) is a kernel theory unifying human, biological, and cognitive intelligence [44]. It is transdisciplinary since it transcends different intelligence disciplines. It unifies knowledge from different disciplines of computing, AI, psychology, and cognitive science. Transdisciplinary principles applied to create theory are unifying knowledge, dealing with socially relevant issues, and transcending disciplines [46]. The theory deals with socially relevant issues by using human intelligence.

The core of UIT theory is abstract intelligence, consisting of a mechanism of models and types of intelligence components and their interactions. The interactions are either direct or indirect through other components. Abstract intelligence provides mechanisms for advanced intelligence, for example, thinking, inference, and perception [44]. The UIT has functional, logical, cognitive, and neural models. Natural intelligence interacts with abstract intelligence through AI. Computational intelligence interacts with abstract intelligence through machinable intelligence. Also, UIT is a design theory that has a defined architecture and environment for intelligent systems carrying out transformations and implementing mechanisms of phenomena [23] in all its components. Humans and intelligent systems are environmental components that carry out transformation. However, the theory does not entirely explain how it is accomplished. Its architecture provides the structure and interaction between its components.

Mathematical intelligence structures (MIS) theory provides a mathematical framework for natural, biological, and artificial intelligence, aiming to collapse the distinction between human brains and artificial mechanisms [47]. Its mathematical formulation makes it easy to apply simulation. However, it primarily focuses on

human intelligence, with limited integration of other biological or non-biological intelligence. It is not clear how non-biological intelligence can be integrated into the three kernel theories.

3.1.1. Kernel theories and CommonKAD

The UIT theory can help understand the types of knowledge and perspectives necessary for developing knowledge-based systems (KBS) and capture different kinds of expertise. Tasks required for an organization to function can be carried out by humans, IS, and AI systems with the necessary expertise and capabilities. The MIS can provide an interface for humans and artificial systems to work together in developing a system using CommonKAD. Also, IDMT can support the decision-making needed to create expert systems with CommonKAD.

3.2. Task theories

Two task-based theories are principles task theory (PTT) [48] and classification task theory (CTT) [49]. These theories focus on dividing the activities required for developing a computing system into tasks, the smallest atomic units of work assignable to a team member. The PTT supports the abstraction and concretization of tasks, differentiating between them, determining complexity, estimating required resources, and constructing tasks [48]. It is based on three task mechanisms: casual relations, task difficulty, and level of detail [50]. Activities are divided into tasks, establishing relationships between those tasks, while principles are the basis of reasoning that guides how actions are executed to perform tasks.

Furthermore, CTT supports AI system development by classifying tasks based on their roles in constructing a system. The theory is based on generic operations required to construct a system by considering input, output, and process by asking what problems the system can solve in mechanical, electronic, and biological systems [51]. The classification is based on system theory: input, output, and process. The tasks are classified into two categories, interpret and construct, each of which is divided into subcategories [49]. Interpret involves identifying, predicting, and controlling while construct involves specifying, designing, and assembling.

The PTT is a partial design theory because it focuses primarily on mechanism and environmental transformational components. Also, CTT has mechanism and environmental transformational components, making it a partial design theory as well.

3.2.1. Task theories and CommonKAD

The CTT can help understand how best to perform the operations required in developing a KBS. Human operations transform different types of knowledge to tackle tasks for creating a KBS. Design strategy transforms AI knowledge,

experience, and perspectives innovatively and continuously into AI products [28]. The various tasks for developing a KBS require different types of knowledge and perspectives.

Moreover, PTT can assist in reasoning about development and knowledge transformation activities, helping improve the CommonKAD task model step by applying task principles. Task theories can guide CommonKAD task model development. The two theories lack ways of dealing with the capabilities to perform tasks required for developing a CommonKAD task model. Capabilities are necessary for task theories because system development cannot take place if the required capabilities are not available.

3.3. Innovation theory

Creativity is a fundamental feature of human intelligence that is challenging for AI to replicate [52]. Machines are still lagging behind human creativity, invention, and innovation in most cases. However, in some areas, AI creativity and innovation are useful in specific contexts. The synergy between AI and human creativity and innovation can be leveraged through human–AI cocreation systems.

AI innovation theory consists of three innovation theories: combinational, transformational, and exploratory [53]. Combinatorial innovation combines existing elements in new ways. Transformational innovation transforms concept space. Transformational innovation applies when concept space is insufficient for innovation [54], which requires adding, dropping, and modifying concepts and relationships to make the concept space sufficient. Exploratory innovation involves exploring the innovation landscape to discover new elements required for innovation.

The Boden theory was used to develop a creative problem-solving framework [54] for creating innovative artifacts. This framework involves problem formulation through planning and learning, with knowledge represented using both symbolic and non-symbolic formalisms. The concept of space is manipulated using various innovation theories. Researchers investigated how to integrate Boden's innovation theories into this framework and ultimately created it to facilitate innovative thinking and problem-solving.

3.3.1. Innovation theory and CommonKAD

Innovation theories can model the evolution of CommonKAD, assist in post-mortem analyses, and identify novel elements for future projects. Artifacts, knowledge produced, and processes used are possible sources for mining novel features. Many software professionals and practitioners know that the methodology they adopt is not the whole truth, so they only adopt parts of it [55]. Applying innovation theory can aid in understanding which parts of CommonKAD to adopt and which parts could be improved for projects.

3.4. Artificial intelligence decision design theory

Crowdsourcing system design theory combines collective intelligence (CI) and AI to support decision-makers in evaluating large amounts of user-generated content [56]. Crowdsourcing is a way of getting ideas from a distributed system of people. Design theory is being used in AI and AI in design, and both to improve systems [57]. Design theory is for developing innovative products.

CI is an emergent property of a distributed system of people using their combined synergized intelligence. Motivation and trust are required by humans to apply CI to design intelligent systems like robots [58]. Most AI systems are based on human individual intelligence or derived intelligence of individuals or communities of organisms, like in ant colony algorithms. Human individuals of a CI system interacting with each other form an adaptive system [58] whose properties cannot be understood by what individuals do but rather by the total sum of their actions.

3.4.1. Decision theory and CommonKAD

Decision-making is core to KBS development. The theory can help enhance CommonKAD by integrating CI into the methodology, systematizing decision-making, and converting tacit knowledge into explicit knowledge. CI is greater than the sum of individual intelligence, making it better for developing KBS as well as being captured in AI knowledge basis.

3.5. Genetic programming theory

Genetic theory is a genetics bioinspired design theory. It is based on schemata theory [59] and evolutionary algorithms [60] that subdivide the search space into subset spaces called schemata [59]. Schemata subspace programs are evolved using generic operators of selection, crossover, and mutation. Selection determines solutions to retain by eliminating a few of the worst solutions from the solution population. Crossover combines parts of existing solutions to create new solutions while mutation generates random novel solution parts. Genetic algorithms have been used to design and model markets, innovation systems, and other types of systems.

3.5.1. Genetic programming theory and CommonKAD

The process of developing KBS innovations can be modeled as a genetic process, considering different perspectives and knowledge types. Genetic programming crossover operations can combine existing elements in new ways, and mutations introduce novel elements. Mutation can be used to complete an innovation concept space driven by AI transformation theory. Genetic programming and combinatorial innovation can be used to combine different knowledge elements in CommonKAD tasks that combine elements.

3.6. Ways and challenges of integrating theories and CommonKAD

One loose way of integrating theories and CommonKAD is applying theories at the beginning of each CommonKAD phase. Another way of loosely integrating them is letting developers decide when to apply theories during development processes. Loosely integrating theories increases the range of problems they can solve. Alternatively, tight integration has the advantage of increasing developer productivity and the disadvantages of reducing the number of problems the combination can solve and the creativity developers can leverage because tight combination leaves little room for flexibility.

CommonKAD is a heavyweight, plan-based methodology that emphasizes comprehensive planning, analysis, design, and documentation. Integrating theories with CommonKAD has both advantages and disadvantages. For example, it may increase overhead costs like project costs beyond what is allowed by small-to-medium AI systems' project budgets [61]. One way to overcome this is by applying the modeling principle of traveling light by creating only minimum viable models that make it easier and faster to create software [62]. Another way is by downscaling CommonKAD to make it lightweight, reducing overheads in developing small-to-medium AI systems.

CommonKAD is modeling-oriented, whereas theories are not. Integrating theories and methodology would require orienting theories towards modeling to create a seamless development process, avoiding the need to switch between modeling and non-modeling orientations. Switching between orientations makes development difficult, requiring more effort and consequently increasing development costs.

Combining theories into a system of theories would create a more powerful way of understanding and improving CommonKAD methodology and other AI artifacts. There are useful approaches combining computing into a system of theories and methods that create a theory nexus [63] and a core design theory based on common field elements [33]. A CommonKAD theory nexus would allow adding more AI and SE methods to the nexus. SE agile methods are the most commonly combined methods in full or partial phases with AI methods for developing AI-based systems [64]. A theory nexus is a loose combination that enables the application of parts of one or more methods on task by leveraging their strengths and counteracting their weaknesses. This gives innovators more freedom to be creative. Nexus solves the problem of having to tackle tasks in fixed ways. Design nexus can be made more innovative by integrating it with AI innovation theories. Methods follow design theories [36, 65]. Among design considerations for creating a design nexus, design theories that methods can follow synergistically are included.

4. Discussion

The theories analyzed in this study involve bioinspired designs that meet some criteria for design theories. Design theories consist of mechanisms, architectural components and their relationships, and environmental transformational components [23]. However, they are often incomplete, either lacking one or more of these three constituents or having incomplete elements. The AI theories considered are based on different foundations and perspectives such as creativity, mathematics, atomicity, genetics, decision theory, and hybrid AI and human CI. Bioinspired designs are considered as an emerging approach combining biology and computing techniques to create novel and useful algorithms. These theories span across multiple disciplines, domains, components, and tasks, reflecting their transdisciplinary nature.

Some of the theories can detail others, overlap with them, or even embed within one another. Like some SE methods, they may overlap and share many elements [55]. AI kernel theories cover nearly all intelligence phenomena and can embed the other types of theories. Task theories can be embedded in other types of theories, detailing theories they get embedded in since a task is the smallest atomic work unit for developing AI artifacts. Many theories have been proposed, including developing transdisciplinary AI theories. Johnson [11] concludes that each of the paradigms and theories has its focus and limitations, and the selection of theories should be based on context. Combining theories and methods in loose combinations like design nexus allows for the application of the most appropriate theory and method to a specific task, and methods have more than one theory they can follow.

A theory consists of a set of concepts and relationships between the concepts. Methods consist of concepts, techniques, and best practices. Design theories can be used to generate design methods [36]. The AI theories analyzed can be used to generate AI methods that are easier to use than the theories they are generated from. The AI methods can follow design theories, and frequently, methods can be used the same as theories [36, 63]. Furthermore, methods following design theories and design theories generating methods can be used to improve existing AI methods like CommonKAD by generating method elements to improve and advance the method. Adapting a method to create a theory is likely to carry over method biases into the theory and present reality the way we think of it rather than as it is [65]. One way to combine AI methods and theories is by establishing guidelines on how methods follow theories during the development of artifacts. Design nexus or methods following theories can help artifacts counteract biases in methods. Although methods and theories perform similarly, both approaches have their limitations.

However, to creatively implement and use SE methodologies, organizations need to embrace a learning culture, be open to new ideas, and encourage experimentation.

This can be improved by adopting the SEMAT framework [55]. The AI design theories' roles in SE methods should be similar to those of SEMAT theory. AI design theories can support a learning culture and experimentation in organizations using AI methodologies such as CommonKAD.

The "SE method wars" have been going on for 50 years, and the evolution of methodologies has followed a zigzag path [55]. Although these method wars are not unique to SE, they are more intense in SE due to its extensive array of technology development methods, potentially more compared to other computing disciplines like AI. The wars divert researchers and practitioners from focusing on real issues, leading to wasted energy in defending methodologies based on biases rather than effectiveness. The zigzag path is inefficient and ineffective, wasting resources and reducing productivity. Using methods that follow theories and creating a design nexus could mitigate these conflicts by reducing biases and providing a more integrated approach.

5. Conclusion

AI theories, as analyzed in this study, are nascent design theories that require more application to generate sufficient evidence. The process of applying these theories not only tests their validity but also provides valuable lessons that could lead to the development of better theories. In scientific fields, theories are often discarded through falsification, while in engineering, they may be deemed not useful after extensive use demonstrates their inadequacy.

Analyzing CommonKAD methodology with AI theories has led to a better understanding and discovery of possible ways of improving it—by introducing new novel perspectives and asking novel questions.

Researchers need to understand the nature of theories within their field to select the most suitable theory for their research and practice and to identify theory gaps. Practitioners can choose the theories that align with their methods as appropriate. Additional research is needed to explore AI theories not covered in this study, focusing on which AI theories have been applied or could have been applied to published research. This can lead to a better understanding of the relevance of AI theories in research and potentially increase their application, leading to theory improvement and the creation of new theories to address identified gaps. Furthermore, more research is necessary to combine and integrate theories to establish a theory nexus and core theories.

Conflict of interest

The author declares no conflict of interest.

References

1 Holmström J. Theorizing in IS research: what came before and what comes next? *Scand J Inf Syst*. 2005;**17**(1):167–174.

2 Johnson P, Jacobson I, Goedicke M, Kajko-Mattsson M. 2nd SEMAT workshop on a general theory of software engineering (GTSE 2013). In: *2013 35th International Conference on Software Engineering (ICSE)*. Piscataway, NJ: IEEE; 2013. p. 1525–1526.

3 Bunge M. Systemism: the alternative to individualism and holism. *J Socio-Econ*. 2000;**29**(2):147–157.

4 Poole DL, Mackworth AK. *Artificial intelligence: foundations of computational agents*. Cambridge: Cambridge University Press; 2010.

5 Shaw M, Garlan D. *Software architecture: perspectives on an emerging discipline*. Hoboken, NJ: Prentice-Hall, Inc.; 1996.

6 Charniak E, McDermott D. *Introduction to artificial intelligence*. Hoboken, NJ: Pearson Education; 1985.

7 Popper K. Science as falsification. In: *Conjectures and refutations*. London: Informa; 1963. p. 33–39.

8 Kuhn TS. *The structure of scientific revolutions*. 2nd ed. Chicago, IL: The University of Chicago; 1970.

9 Lakatos I. Falsification and the methodology of scientific research programmes. In: *Criticism and the growth of knowledge*. Cambridge, MA: Cambridge University Press; 1970. p. 91–195.

10 Minsky M. Commonsense-based interfaces. *Commun ACM*. 2000;**43**(8):66–73.

11 Johnson C. *What is research in computing science? [Internet]*. Glasgow: Computer Science Dept., Glasgow University; 2006. Electronic resource: https://www.dcs.gla.ac.uk/~johnson/teaching/research_skills/research.html.

12 Mambo WN. Aligning software engineering and artificial intelligence with transdisciplinary. *Transdiscipl J Eng Sci*. 2022;**13**: 21–34.

13 IEEE. *IEEE Standard glossary of software engineering terminology*. Piscataway, NJ: IEEE; 1990.

14 Collins C, Dennehy D, Conboy K, Mikalef P. Artificial intelligence in information systems research: a systematic literature review and research agenda. *Int J Inf Manag*. 2021;**60**: 102383.

15 Schreiber G, Wielinga B, de Hoog R, Akkermans H, Van de Velde W. CommonKADS: a comprehensive methodology for KBS development. *IEEE Expert*. 1994;**9**(6):28–37.

16 Avison D, Fitzgerald G. *Information systems development: methodologies, techniques and tools*. New York, NY: McGraw-Hill; 2003.

17 Jacobson I, Spence I, Ng P. Agile and SEMAT—perfect partners. *Commun ACM*. 2013;**56**(11):54–61.

18 Schiaffonati V, Verdicchio M. The influence of engineering theory and practice on philosophy of AI. In: *Philosophy and theory of artificial intelligence*. Cham: Springer; 2013. p. 375–388.

19 Nicolescu B. Methodology of transdisciplinarity: levels of reality, logic of the included middle, and complexity. *Transdiscipl J Eng Sci*. 2010;**1**(1):19–38.

20 Franssen M, Lokhorst G, Van de Poel I. Philosophy of technology. In: *The Stanford encyclopedia of philosophy*. Stanford, CA: Stanford University; 2018.

21 Hevner AR, March ST, Park J, Ram S. Design science in information systems research. *Manag Inf Syst Q*. 2004;**28**(1):75–105.

22 Holz HJ, Applin A, Haberman B, Joyce D, Purchase H, Reed C. Research methods in computing. In: *Working group reports on ITiCSE on Innovation and technology in computer science education*. New York, NY: ACM; 2006. p. 96–114.

23 Wieringa R, Daneva M, Condori-Fernandez N. The structure of design theories, and an analysis of their use in software engineering experiments. In: *2011 International Symposium on Empirical Software Engineering and Measurement*. Piscataway, NJ: IEEE; 2011. p. 295–304.

24 Vaishnavi V, Kuecher W. *Design research in information systems*. Cham: Springer; 2009.

25 Muller VC. Introduction: philosophy and theory of artificial intelligence. *Minds Mach*. 2012;**22**(2):67–69.

26 Bonsignorio F. The new experimental science of physical cognitive systems: AI, robotics, neuroscience and cognitive sciences under a new name with the old philosophical problems? In: *Philosophy and theory of artificial intelligence*. Cham: Springer; 2013. p. 133–150.

27 Davenport D. The two (computational) faces of AI. In: *Philosophy and theory of artificial intelligence*. Cham: Springer; 2013. p. 43–58.

28 Lindberg T, Meinel C, Wagner R. Design thinking: a fruitful concept for IT development? In: *Design thinking: understand–improve–apply*. Cham: Springer; 2011. p. 3–18.

29 Wang P. *Theories of artificial intelligence: meta-theoretical considerations*. Cham: Springer; 2012. p. 4654–4665.

30 Rogers EM. *Diffusion of innovations*. New York: Free Press; 1983.

31 Hodges P, Ruecker S, Scaletsky C, Rivera J, Faller R, Geppert A. Four criteria for design theories. *She Ji: J Des Econ Innovation*. 2017;**3**(1):65–74.

32 Sjøberg DI, Dybå T, Anda BC, Hannay JE. Building theories in software engineering. In: *Guide to advanced empirical software engineering*. Cham: Springer; 2008. p. 312–336.

33 Jacobson I, Ng PW, McMahon PE, Spence I, Lidman S. The essence of software engineering: the SEMAT kernel. *Commun ACM*. 2012;**55**(12):42–49.

34 Ralph P. The sensemaking-coevolution-implementation theory of software design. *Sci Comput Program*. 2015;**101**: 21–41.

35 Tams S. On the appropriateness of theory borrowing in IS: an interdisciplinary evaluation. In: *Proceedings of the Southern Association for Information Systems Conference*. Atlanta, GA: AISNET; 2010. p. 130–135.

36 Hall JG, Rapanotti L. A design theory for software engineering. *Inf Softw Technol*. 2017;**87**: 46–61.

37 Kannengiesser U, Gero JS. Is designing independent of domain? Comparing models of engineering, software and service design. *Res Eng Des*. 2015;**26**: 253–275.

38 Gero JS, Kannengiesser U. The situated function–behaviour–structure framework. *Des Stud*. 2004;**25**(4):373–391.

39 Stol KJ, Fitzgerald B. Theory-oriented software engineering. *Sci Comput Program*. 2015;**101**: 79–98.

40 Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in software engineering*. Cham: Springer; 2012.

41 Mariotti S. Forging a new alliance between economics and engineering. *J Ind Bus Econ*. 2021;**48**(4):551–572.

42 Xie T. Intelligent software engineering: synergy between AI and software engineering. In: *Proceedings of the 11th Innovations in Software Engineering Conference*. New York, NY: ACM; 2018.

43 Gruner S. Invited lecture: notions of 'theory' and their practical consequences in the discipline of software 'engineering' (including Information Systems Design). *S Afr Comput J*. 2020;**32**(2):293–322.

44 Wang Y. On abstract intelligence: toward a unifying theory of natural, artificial, machinable, and computational intelligence. *Int J Softw Sci Comput Intell*. 2009;**1**(1):1–17.

45 Hutter M. Towards a universal theory of artificial intelligence based on algorithmic probability and sequential decisions. In: *European Conference on Machine Learning*. Cham: Springer; 2001. p. 226–238.

46 Pohl C. From transdisciplinarity to transdisciplinary research. *Transdiscipl J Eng Sci*. 2010;**1**: 65–73.

47 Panchariya DA. The theory of natural-artificial intelligence. *Eur J Artif Intell Mach Learn*. 2022;**1**(1):1–3.

48 Thórisson KR, Bieger J, Thorarensen T, Sigurðardóttir JS, Steinbrink BR. Why artificial intelligence needs a task theory: and what it might look like. In: *AGI 2016 Proceedings 9*. Cham: Springer; 2016. p. 118–128.

49 Clancey WJ. Heuristic classification. *Artif Intell*. 1985;**27**(3):289–350.

50 Belenchia M, Thórisson KR, Eberding LM, Sheikhlar A. Elements of task theory. In: *Artificial General Intelligence: 14th International Conference*. Cham: Springer; 2022. p. 19–29.

51 Jackson P. *Introduction to expert systems*. 3rd ed. Boston: Addison-Wesley; 1999.

52 Boden MA. Creativity and artificial intelligence. *Artif Intell*. 1998;**103**(1–2):347–356.

53 Boden MA. Computer models of creativity. *AI Mag*. 2009;**30**(3):23–23.

54 Gizzi E, Nair L, Chernova S, Sinapov J. Creative problem solving in artificially intelligent agents: a survey and framework. *J Artif Intell Res*. 2022;**75**: 857–911.

55 Jacobson I, Stimson R. Escaping method prison—on road to real software engineering. In: *Essence of software engineering*. Cham: Springer; 2018.

56 Rhyn M, Blohm I. Combining collective and artificial intelligence: towards a design theory for decision support in crowdsourcing [Internet]. In: *Proceedings of the 25th European Conference on Information Systems (ECIS), Guimarães, Portugal*. 2017. p. 2656–2666. Available from: https://aisel.aisnet.org/ecis2017_rip/18.

57 Neuhauser L, Kreps GL, Morrison K, Athanasoulis M, Kirienko N, Van Brunt D. Using design science and artificial intelligence to improve health communication: ChronologyMD case example. *Patient Educ Couns*. 2013;**92**(2):211–217.

58 Salminen J. Collective intelligence in humans: a literature review [Internet]. In: *Proceedings Collective Intelligence Conference (CI)*. 2012. Available from: https://arxiv.org/abs/quant-ph/1204.2991.

59 Vanneschi L, Poli R. Genetic programming: introduction, applications, theory and open issues. In: *Handbook of natural computing*. Berlin: Springer; 2011. p. 709–739.

60 Holland J. *Adaptation in natural and artificial systems*. Cambridge, MA: MIT Press; 1975.

61 Surakratanasakul B. Lightweight commonKAD in knowledge intensive organization. In: *ICITEE*. Piscataway, NJ: IEEE; 2017.

62 Pressman R. *Software engineering: a practitioner's guide*. New York, NY: McGraw-Hill; 2010.

63 Pries-Heje J, Baskerville R. The design theory nexus. *Manag Inf Syst Q*. 2008;**32**(4):731–755.

64 Syahputri IW, Ferdiana R, Kusumawardani S. Does system based on artificial intelligence need software engineering method? Systematic review. In: *2020 Fifth International Conference on Informatics and Computing (ICIC)*. Piscataway, NJ: IEEE; 2020.

65 Ralph P. Developing and evaluating software engineering process theories. In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. vol. 1, Piscataway, NJ: IEEE; 2015. p. 20–31.