www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

AN INNOVATIVE SOFTWARE DEVELOPMENT METHODOLOGY FOR DEEP LEARNING-DRIVEN VISUAL COMPUTING IN BUILT ENVIRONMENT APPLICATIONS

SUBMITTED: September 2024
REVISED: April 2025
PUBLISHED: June 2025
EDITORS: Yang Zou, Mostafa Babaeian Jelodar, Zhenan Feng, Brian H.W. Guo
DOI: 10.36680/j.itcon.2025.041

Prasad Perera, Doctoral Researcher
Centre for Smart Modern Construction, Western Sydney University, Kingswood, Australia
ORCID: https://orcid.org/0000-0002-8571-0196
prasad.perera@westernsydney.edu.au
Srinath Perera, Professor
Centre for Smart Modern Construction, Western Sydney University, Kingswood, Australia
ORCID: https://orcid.org/0000-0001-9218-9282
srinath.perera@westernsydney.edu.au
Xiaohua Jin, Associate Professor
Centre for Smart Modern Construction, Western Sydney University, Kingswood, Australia
ORCID: https://orcid.org/0000-0003-0877-9533
xiaohua.jin@westernsydney.edu.au
Maria Rashidi, Associate Professor
Western Sydney University, Kingswood, Australia
ORCID: https://orcid.org/0000-0003-2847-3806
m.rashidi@westernsydney.edu.au
Samudaya Nanayakkara, Lecturer
Centre for Smart Modern Construction, Western Sydney University, Kingswood, Australia
s.nanayakkara@westernsydney.edu.au
Gina Yazbek, CEO
Commnia Pty Ltd, Sydney, Australia
gina@commnia.com
Andrew Yazbek, BDM
Commnia Pty Ltd, Sydney, Australia
andrew.yazbek@commnia.com

SUMMARY*: This paper presents an innovative software development methodology, the GENESIS (Generalised ENgineering for Embedded Software with Integrated AI System) Methodology, tailored for Deep Learning (DL)-driven visual computing applications in the built environment. Integrating AI into embedded systems has presented unique challenges to the associated software development methodologies. The proposed GENESIS Methodology integrates Design Science Research principles with established Artificial Intelligence (AI) embedded software-specific software engineering practices. Further, the approach has co-opted and synthesised insights from recent studies on AI software development and software engineering methodologies, incorporating key elements. The GENESIS Methodology encompasses twelve key stages, from problem definition to monitoring and maintenance for the developed software systems, with the sharing of knowledge, focusing on data-centric development and model-driven AI approaches. The systematic integration of AI-specific software engineering stages within conventional software engineering methodology uniquely combines a research-driven approach. The emphasis on the importance of Convolutional Neural Networks (CNNs) for visual computing tasks details the technical considerations for training and evaluating Deep Learning models. The paper justifies adopting the Waterfall model for its structured approach, aligning with the needs of the technically complex systems. Finally, a software prototype development is presented using the proposed GENESIS Methodology, and the functionality is focused on the built environment, validated by achieving a 91.2% accuracy in identifying six types of concrete defects, demonstrating the accuracy of this approach in real-world applications. This comprehensive methodology aims to enhance the development of AI-based visual computing applications in the built environment, offering a systematic framework.*

KEYWORDS*: Software Engineering Methodology, Deep Learning, Visual Computing, Built Environment, AI-Embedded Systems.*
REFERENCE*: Prasad Perera, Srinath Perera, Xiaohua Jin, Maria Rashidi, Samudaya Nanayakkara, Gina Yazbek & Andrew Yazbek (2025). An Innovative Software Development Methodology for Deep Learning-Driven Visual Computing in Built Environment Applications. Journal of Information Technology in Construction (ITcon), Special issue: 'Construction 5.0', Vol. 30, pg. 1017-1040, DOI: 10.36680/j.itcon.2025.041*

1. INTRODUCTION

The rapid advancement of Digital Technologies, especially Artificial Intelligence (AI) technologies, has opened new frontiers in various domains, including the built environment (Bier *et al.* 2024; Harichandran *et al.* 2023; Rampini and Re Cecconi 2022; Samuelson and Stehn 2023; Wang J *et al.* 2024). Further, the Construction 4.0 revolution has created a thrust on the development of software prototypes for multiple applications (Doukari *et al.* 2022; Huang Qian *et al.* 2019; Perera *et al.* 2024; Perrier *et al.* 2020; Singh and Anumba 2022; Zhou X and Flood 2024). The availability of affordable, powerful computing and communication hardware has increased the technical feasibility of the deployment of such AI-powered systems (Heyn *et al.* 2021). Visual computing, powered by Deep Learning algorithms, has enabled the development of multiple applications for enhancing design, construction, and maintenance processes in architecture, engineering, and construction (AEC) industries (Ekanayake *et al.* 2021; Eliwa *et al.* 2023; Jiang Z *et al.* 2023; Perera P *et al.* 2023; Uggla and Horemuz 2020). However, integrating these advanced technologies into practical applications presents unique challenges, particularly when the software systems are developed by integrating AI-embedded software modules (Heyn *et al.* 2021; Martin *et al.* 2022; Nilsson *et al.* 2024).

A software development methodology or software engineering process is a systematic development process focused on the product and the process (Phillip A. Laplante 2022). It could be defined as the engineering discipline concerned with the software development lifecycle, having defined processes and methods (Linares-Vásquez 2024; Tsui *et al.* 2022). Software engineering has expanded beyond the regular software systems to AI-embedded systems, Quantum Software Engineering and Super Computer-driven Systems (Akbar *et al.* 2024; Clement *et al.* 2023; Russo 2024; Treude 2023; Zeb *et al.* 2023; Zhao 2020). The AI-embedded software systems require specific software engineering methodologies (Beneder and Schmitt 2024; Dakkak *et al.* 2024). Traditional software engineering practices do not meet the complex methodological requirements when applied to AI-embedded systems (Giray 2021a, 2021b; Saoudi *et al.* 2023; Vermesan *et al.* 2022). The complexity of AI models, the complex data requirements and the experimental nature of such software systems require innovative software engineering techniques (Giray 2021a; Sanic *et al.* 2022). The importance of the human-centric nature of AI-embedded systems is highlighted by Islam (2021), arguing on the fact that AI systems should be both trustworthy and designed with a human-centric approach, ultimately for policymakers to validate the application of such software systems.

Regular software engineering methodologies have derived conventional methods to capture requirements, design the software, test and deploy (Hewavitharana *et al.* 2025). However, AI-embedded software systems require specific software development methodologies since there are specific software design, development and testing methods derived (Giray 2021a). Further, ensuring the security and reliability of the software systems embedded with AI technologies could be achieved with specific forensic methodologies and techniques (Wang C *et al.* 2023). In light of this context, a thorough review of existing conventional software development methodologies is performed, followed by an examination of their incorporation into AI-embedded software systems focused on the built environment. This study has focused on an innovative and integrative software development methodology specifically tailored for Deep learning- driven visual computing applications in the built environment.

The proposed GENESIS (Generalised ENgineering for Embedded Software with Integrated AI System) Methodology integrates and co-opts insights from recent studies on AI software development, including works by McDermott *et al.* (2021), Llinas *et al.* (2021), Amershi *et al.* (2019), (Zhou Y *et al.* 2020) and guidelines by Horneman *et al.* (2020) et al. Furthermore, the need for a research-based approach to AI-embedded software development is highlighted. This is necessitated by the fact that problem identification and validation are conducted as research to ensure the validity of the software system's purpose. AI-specific considerations are integrated with the principles of Design Science Research, creating a comprehensive framework that addresses the unique challenges of developing AI-embedded systems. Further, complex and research-oriented software systems with AI-embedded modules focused on the applications in the built environment face challenges with standardised AI integration strategies, difficulty in lifecycle management, and inefficiencies in AI-specific development workflows. To address these gaps, this study introduces the GENESIS Methodology, a structured software development framework that integrates AI-specific software engineering principles with Design Science Research frameworks.

The proposed GENESIS Methodology encompasses twelve key stages, from problem definition to monitoring and maintenance for the developed software systems, and finally up to the stage of knowledge sharing with particular

emphasis on data-centric development and model-driven AI approaches. The systematic integration of AI-specific software engineering stages within conventional software engineering methodology uniquely combines a research-driven approach to capture complex requirements and to involve domain experts in creating an innovative approach towards AI-embedded software development. The systematic software development methodology focuses on the importance of Convolutional Neural Networks (CNNs) for visual computing tasks and provides detailed technical considerations for training and evaluating Deep Learning models (Tsai and Wu 2024). The adoption of the Waterfall model as the foundational software development approach is justified by its alignment with the high-performance and complexity demands of systems, following a thorough comparative analysis of different models and their suitability for AI-embedded systems (Alevizos *et al.* 2024). To test and validate the proposed GENESIS methodology, a software prototype was developed featuring an AI-embedded system designed to detect cracks in concrete, and the resulting outcomes are presented. The key contribution to knowledge lies in the detailed, integrated approach of GENESIS, tailored to AI-driven systems, exemplified by the development of AI-based visual computing applications for the built environment.

The structure of the paper follows with Section 2 providing the background of the study through a comprehensive literature review on AI software development methodologies and visual computing in the built environment, Section 3 details the proposed methodology, including its theoretical foundation and practical implementation, Section 4 presents a case study demonstrating the application of the methodology in a real-world scenario and Section 5 discusses the implications of the approach and its potential impact on the field. Finally, Section 6 concludes the paper and outlines directions for future research.

2. LITERATURE REVIEW

Research methodology is the systematic approach used to collect, analyse, and interpret data in order to answer research questions or test hypotheses. It includes principles, procedures, and tools used to conduct research in a particular field. There are different research methodologies used, and the choice of methodology depends on the research question being investigated and the type and amount of data that needs to be collected (Sutrisna 2009). The development of AI-embedded software systems focused on a specific application or requirement can be considered part of a research process, as the development methodology and the required dataset must be meticulously researched to ensure the highest accuracy of the automated application (Kakatkar *et al.* 2020). Moreover, the software development, conducted as proof of the GENESIS methodology, was guided by a proper research methodology to create a valid artifact. This approach aligns with the need for rigorous methodological frameworks in AI-embedded software development, as highlighted in recent literature (Barenkamp *et al.* 2020; Batarseh *et al.* 2020; Kulkarni and Padmanabham 2017). . The design science research methodology (DSRM) encompasses the fundamental principles, practices, and procedures essential for conducting such research and achieves three primary objectives. DSRM is distinct from other paradigms, including theory-building, testing and interpretive research, in that it emphasises the creation of solutions that serve human purposes, rather than simply understanding reality as natural and social sciences attempt to do. Moreover, in DSRM research, the design and demonstration of the practical usefulness of an engineering or computer science-based solution is regarded as the central element of the methodology (Peffers *et al.* 2007). The alignment of the software engineering process of an innovative solution through the use of DSRM is elaborated through the review study conducted by Engström *et al.* (2020) validating the approach. Drechsler and Hevner (2016) elaborates on the extensive utilisation of DSRM in information systems research, which has been established as a prominent research paradigm. They also emphasise the significance of building upon existing literature and propose a mental model for assessing design science research within the information systems context.

According to Heroux (2023) recognising the approach of Research Software Science (RSS) is of utmost importance, as it involves the application of scientific methods to comprehend and enhance the development and utilisation of software for research purposes. RSS necessitates the utilisation of systematic observation and experimentation to acquire and disseminate knowledge via repeatable and reproducible processes, accomplished through the implementation of a software prototype. Reinforcing the concept Zhang *et al.* (2016) and Giray (2021a) had emphasised the need to follow a research-based approach for innovative technologies to be used in applications. Conboy *et al.* (2021) had extended the concept by conducting a comprehensive literature review on the state-of-the-art research related to AI-embedded software. Through his research, it was demonstrated that there exist five essential criteria for conducting high-quality AI research, which ultimately lead to the development of effective AI-embedded software systems.

Martínez-Fernández *et al.* (2022) defined AI-embedded systems as software systems that incorporate at least one AI component. Due to the advancements in AI technology, these systems are becoming increasingly feasible for application deployment. However, there is a lack of comprehensive knowledge regarding Software Engineering (SE) practices for developing, operating, and maintaining AI-based systems (Khomh *et al.* 2018; Linares-Vásquez 2024). Further, the researchers argue that the specific Software Engineering Methodologies focused on the development of AI-integrated systems are yet to mature. Giray (2021a) strengthened the argument by highlighting recent findings that a significant proportion, specifically 47%, of AI initiatives are unable to progress beyond the prototype stage due to the absence of adequate tools for creating and sustaining production-grade AI systems.

The software development methodologies defined for AI-embedded software development require special software engineering techniques to deploy such systems successfully (Beneder and Schmitt 2024; Dakkak *et al.* 2024). Traditional software engineering practices, while robust for conventional systems, often fall short when applied to AI-embedded systems (Giray 2021a, 2021b; Saoudi *et al.* 2023; Vermesan *et al.* 2022). The inherent complexity of Deep Learning models, the critical role of data in their performance, and the need for continuous learning and adaptation necessitate a paradigm shift in how we approach software development for these applications (Munappy *et al.* 2019; Sanic *et al.* 2022). The experimental nature of AI-embedded systems from the start to finish of the development process, heavy data dependency, the requirement of continuous quantitative & qualitative evaluation and continuous monitoring of performance are some of the key features differentiating AI-embedded systems from regular software systems (Giray 2021a).

The transformation of the DevOps methodology towards MLOps in order to incorporate embedded machine learning modules in software systems has transformed the software development industry, focused on AI-embedded software development (John *et al.* 2021; Kreuzberger *et al.* 2023). Kreuzberger *et al.* (2023) argues that even though the motive of MLOps is to deliver ML-embedded software systems efficiently, many development projects have failed due to the vagueness of the methodology. However, MLOps is considered one of the best methodologies developed for feature-specific ML engineering and end-to-end lifecycle management for ML applications (Tamburri 2020; Zhou Y *et al.* 2020). Further, Zhou Y *et al.* (2020)had highlighted the capability of MLOps to manage continuous integration and deployment (CI/CD) pipelines for ML-driven systems. This iterative development capability of MLOps will be considered for the proposed methodology while comparing it with Agile AI Development and DevOps for AI. Figueiredo *et al.* (2025) had studied the effectiveness of integrating DevOps with Agile Methodologies towards the development of AI-embedded systems. The study highlighted the advantages of continuous development and refinement of AI models through agile techniques while following DevOps processes to manage the delivery of the software systems. Lwakatare *et al.* (2020), in their study reinforces the concept of using Agile combined with DevOps methodologies for continuous integration and deployment (CI/CD) pipelines for AI-driven systems. Extensive analysis of the above-mentioned studies highlights the need for Agile methods for continuous integration and deployment required for the AI models to be updated and refined for best performance and applicability for AI-embedded systems, while presenting MLOps as an integrated solution for continuous integration and deployment (CI/CD) pipelines for AI-driven systems.

Martínez-Fernández *et al.* (2022) and Giray (2021a) had based their arguments on the fact that developing, operating, and maintaining AI-based systems differ from traditional software systems as they do not rely on written program code but rather on inferred rules and system behaviour derived from training data. Consequently, interdisciplinary teams of data scientists and software engineers are required to ensure the quality of the system (Kim M *et al.* 2018; Vogelsang and Borg 2019). The design and analysis of AI-based systems need to focus on different quality attributes, such as large and changing datasets, robust and evolutionary infrastructure, as well as ethics and equity requirements, to avoid creating technically deficient systems (Horkoff 2019; Hu *et al.* 2020). Therefore, exploring Software Engineering (SE) practices for the development, maintenance, and evolution of AI-based systems is crucial. Santhanam *et al.* (2019) had argued that the incorporation of an AI component in mission-critical or safety-critical applications could result in unanticipated outcomes given the present level of maturity. Hence, there are significant concerns about the dependability, consistency, credibility, and sustainability of AI applications (Horkoff 2019). With AI's dominance despite its limitations, there is a growing requirement for more structured methodologies towards AI software development and certification (Nascimento EdS *et al.* 2019; Ozkaya I. 2020).

Reinforcing the idea Hossain Faruk *et al.* (2023) had presented the integration of artificial intelligence (AI) has the potential to create a new era and transform the existing software engineering paradigm. The next section will include the detailed methodological development of the proposed GENESIS methodology.

3. SYSTEM DEVELOPMENT METHODOLOGY

The development of the GENESIS Methodology was based on key conceptual differences between traditional software systems and AI-embedded software systems. These concepts were reinforced through the literature review included in this study. The extensive analysis of the existing AI-embedded Software Development Methodologies and the co-opting of such methodologies is presented in the following sections.

3.1 Analysis of Existing AI-Embedded Software Development Methodologies and Concepts

The first conceptual difference is how it interacts with devices/hardware in terms of input, output, and the developed program/system. The following illustration (Figure 1) is inspired by the concept presented by Domingos (2015). It indicates that the interaction of regular programs and AI-embedded software with hardware and devices is conceptually different. Furthermore, AI-embedded systems are highly complex and interdisciplinary, which adds to their complexity.

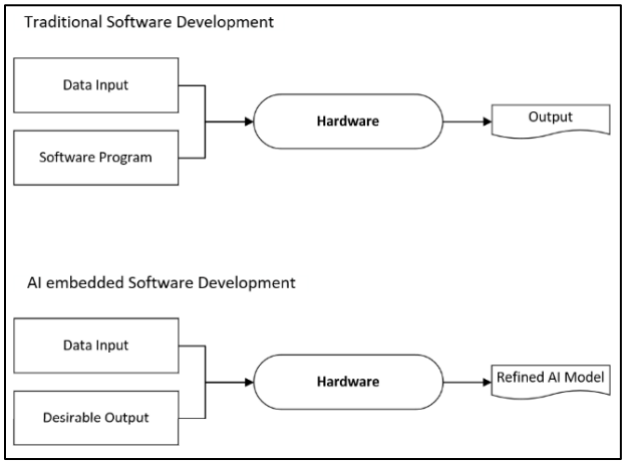

Figure 1: Hardware Interaction Paradigms: Traditional Software Development vs AI Embedded Software Development (Inspired by (Domingos 2015)).

The second conceptual difference is how regular programming and AI-embedded programming generate the outcomes. The amount of human intervention for the two programming paradigms is conceptually different, as illustrated in Figure 2. The traditional software development process requires human intervention in feature extraction and model building, while in AI-embedded software development, the features learnt and the models are created automatically.

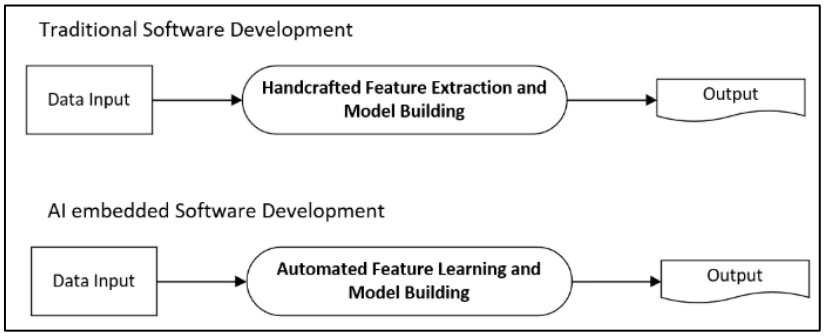

Figure 2: Humancrafting and Automated Paradigms: Traditional Software Development vs AI Embedded Software Development (Inspired by (Janiesch et al. 2021)).

The research work by Hevner *et al.* (2004); Gregor and Hevner (2013); Drechsler and Hevner (2016) had extensively presented the theoretical background of the DSRM and was thoroughly studied to create the rationale for Design Science Research Methodology in software system development. Furthermore, the authors have developed a specific methodology for Design Science Research Methodology in software system development. The rationale for selecting Design Science Research Methodology for software system development is due to the following reasons:

1. Design Science Research Methodology focuses on the identification of real-life problems and resolving them through innovative artifacts.
2. It advocates for an iterative design, evaluation, and refinement process. This is particularly valuable for AI-embedded software systems, allowing for continuous refinement of AI models and algorithms to optimise final accuracy.
3. It promotes a theoretical research-based approach and practical deployment of the suggested technology. This is important since proper algorithms should be researched and relevant data should be developed for practical deployment.
4. It performs a rigorous evaluation of artifacts for their effectiveness and applicability. For AI-embedded software systems, where validation of models and performance is crucial, this rigorous evaluation process is essential.
5. It facilitates the development of new or improved artifacts through its methodology. AI-embedded systems are often complex and novel, requiring a methodology that supports innovation and detailed design processes.

Hevner *et al.* (2004) had developed a DSRM specifically for Information System Development in their studies and Dresch *et al.* (2014) defined different stages of the DSRM for systems development. Further, Hevner *et al.* (2004) defined seven guidelines as steps to be performed in DSRM. These guidelines are correlated to the stages of the DSRM derived by (Dresch *et al.* 2014).

A comprehensive Software Development Methodology is crucial for successfully deploying software systems. The systematic and collaborative process of software development should be aligned with the domain knowledge and the technology for seamless development (Jiang A *et al.* 2022; Paskaleva *et al.* 2023; Tallgren *et al.* 2020).

Martínez-Fernández *et al.* (2022) and Kurrek *et al.* (2020) had presented Software Engineering for Artificial Intelligence (SE4AI), which is one of the latest software development methodologies currently being explored. Further Kurrek *et al.* (2020) had presented the Q-Model, which is an Artificial Intelligence-based software application development methodology, by referring to theories developed.

Santhanam *et al.* (2019) had presented a model consisting of three key elements to consider for AI-embedded software development, further argued that these three factors collectively establish the essential prerequisites, and the absence of any one of them could potentially lead to failure in the corresponding AI-based software systems. The following Figure 3 illustrates the three key success factors of AI-embedded software systems derived from the conclusions of the study.

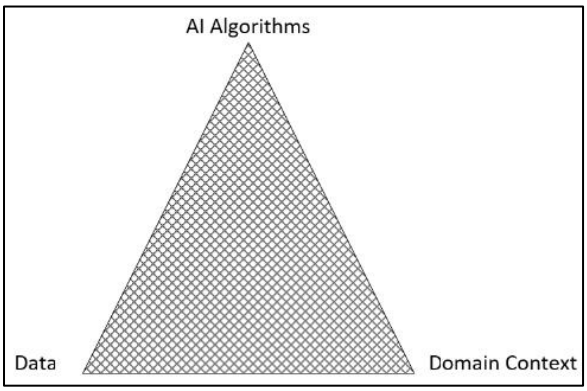

Figure 3: The "Triad of AI": Critical Success Factors (Inspired by ((Santhanam et al. 2019)).

Santhanam *et al.* (2019) elaborated on the concept by establishing equal importance on data for training and validation, selection of the proper algorithm and study of the relevant domain before the development of the AI-embedded information system. Amershi *et al.* (2019) team of researchers from Microsoft conducted a study to evaluate the software engineering process adopted by Microsoft software teams in their development of software applications that incorporate customer-focused AI features. The study presented a foundational model for the development of AI-based information systems. The study further identified three fundamental differences in building applications and platforms with AI components in comparison to regular applications. Firstly, the core of AI components revolves around data. The process of discovering, sourcing, managing, and versioning data involves an inherently more intricate and distinct approach than that of software code. Secondly, building for the customisability and extensibility of models necessitates that teams possess not only software engineering skills but also a deep understanding of AI, enabling them to construct, assess, and refine models from scratch. Thirdly, maintaining strict module boundaries between machine learning components can be more challenging than for software engineering modules. AI components and models can become enmeshed with the regular software functions, causing them to affect one another during the process of training and tuning (Amershi *et al.* 2019).

Llinas *et al.* (2021) had discussed AI/ML software development methodologies regarding several studies conducted collaboratively with NASA, the U.S. Department of Defence (DoD), and the US government. The researchers highlight that the dominant characteristic of AI/ML software development lies in the dependence of the process on data characteristics. The life cycle of the AI/ML model can be characterised as a process that entails dealing with data and selecting a target classification model. The selection of data to learn, and then conditioning that data for the targeted purposes of the application, require a significant amount of domain knowledge and are interconnected and non-linear. Collectively, these steps are known as "Feature Engineering," which is the process of using domain knowledge to extract features from raw data, often via data mining or other techniques.

Ultimately Horneman *et al.* (2020) had developed a set of 11 foundational practices based on research conducted by the DoD on the development of artificial intelligence-enabled systems. The DoD was aiming to enhance the speed and accuracy of decision-making in support of their missions through the improved capabilities of ML algorithms and the availability of abundant computing resources allows the application of these algorithms to models containing thousands of parameters, as well as terabytes and petabytes of data. A comprehensive roadmap was developed by McDermott *et al.* (2021) focused on SE4AI to create human-centric, sustainably scalable, robust, and secure AI-based applications.

It is crucial to comprehend the foundational practices and concepts presented in the conceptual roadmap by the authors to understand the key points necessary for developing an adapted software development methodology for AI-embedded software systems. The eleven foundational principles developed by Horneman *et al.* (2020), specifically designed for the DoD, are as follows, in relation to the roadmap developed for AI-based applications:

1. Identify problems solvable exclusively through AI, determining desired outcomes and necessary data to achieve them. This ensures that AI is applied where it can provide unique value.
2. Assemble multidisciplinary software engineering teams comprising subject matter experts, data scientists, and data architects. This ensures a comprehensive approach to AI development, leveraging diverse expertise.
3. Prioritise data integrity by implementing robust safeguards against malicious attacks and maintaining regular data cleansing protocols. This proactive approach prevents project disruptions due to data-related issues.
4. Select algorithms based on the specific requirements of the model rather than their popularity. Recognize that optimal algorithm choices may evolve as the project progresses and innovations emerge.
5. Implement comprehensive monitoring and mitigation strategies to secure AI systems against potential threats. This involves continuous risk assessment and the development of robust security mechanisms.
6. Establish clear checkpoints throughout the development process to facilitate recovery, ensure traceability, and justify decisions as the project evolves. This supports both project management and system integrity.
7. Integrate user experience (UX) and interaction considerations to continuously validate and evolve models and architecture. Monitor UX metrics for early detection of performance issues, ensuring the system remains user-centric and effective.

8. Design systems to interpret the inherent ambiguity in AI outputs effectively. This involves developing mechanisms to aid in output interpretation and ensure accuracy, acknowledging the probabilistic nature of many AI systems.
9. Develop systems that are loosely coupled, extensible, scalable, and secure. This architectural approach allows for adaptation to inevitable changes in data, models, and algorithmic innovations without necessitating complete system overhauls.
10. Allocate sufficient resources to accommodate constant and enduring change throughout the system's lifecycle. This includes planning for computational resources, hardware upgrades, storage capacity, bandwidth requirements, expertise acquisition, and time allocation for ongoing development and maintenance.
11. Address ethical considerations comprehensively by evaluating every aspect of the system for potential ethical issues. This includes scrutinizing the system's intended use, data representation methods, and model structure to ensure responsible AI development and deployment.

(Horneman *et al.* 2020)

The foundational principles integrated with the existing software development methodologies presented by multiple scholars elaborated above had synthesised a robust framework for developing AI-embedded software systems. Considering the unique challenges and complexities inherent in AI-embedded system development, these frameworks have provided the baseline for the GENESIS Methodology. The adaptation and co-opting of the discussed methodologies and frameworks are used as a base to create more effective, secure, and ethically sound AI systems that are better equipped to meet the complex demands of modern applications for the built environment.

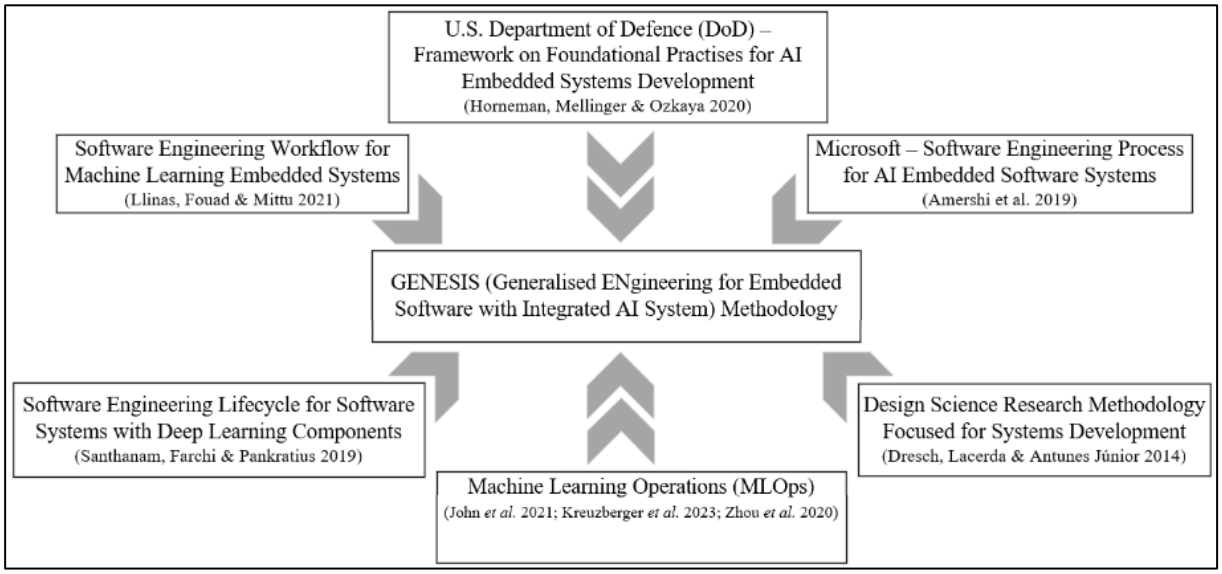

Figure 4: Design Science Research Methodology infused with GENESIS Methodology.

3.2 Proposed GENESIS Methodology through Integration of Existing Methodologies

The GENESIS Methodology was derived through an infusion of software engineering methods presented by McDermott *et al.* (2021), Llinas *et al.* (2021) and Amershi *et al.* (2019) initial models and the AI software development guidelines presented by Horneman *et al.* (2020) was defined to create the fundamentals of an AI software development methodology.

Carstensen and Bernhard (2019) derived and presented processes within the design life cycle that were developed and presented to be integrated into the Design Science Research Methodology, which aims to solve problems through innovation. This methodology encompasses the concepts, practices, technical capabilities, and products used to analyse, design, implement, manage, and utilise information systems. Gestão *et al.* (2021) conducted a

literature study to review the application of Design Science Research Methodology to Information Systems development. The study presented the main steps of the Design Science Research Methodology by analysing the theoretical establishment of Design Science Research for Information System Development.

The AI software development methodology was created through an infusion of software development processes derived from models presented by McDermott *et al.* (2021), Llinas *et al.* (2021), Amershi *et al.* (2019) and Horneman *et al.* (2020), were integrated with the design science research methodology (DSRM). These models were synthesised with the core principles of DSRM to create a comprehensive framework tailored specifically for AI Software Development. The resulting framework, the GENESIS Methodology illustrated in Figure 4, represents a holistic approach that addresses the unique challenges and requirements of AI-embedded systems.

The GENESIS Methodology was developed based on validated theoretical and philosophical foundations, ensuring it meets the technical demands of AI software development while aligning with scientific and ethical considerations. This framework comprises interconnected steps tailored to address specific aspects of AI software development, offering a structured yet flexible approach for navigating complexities. By integrating established methodologies, this Design Science Research Methodology (DSRM) framework equips researchers and developers with a robust tool for creating innovative and ethically sound AI-embedded systems. The following section outlines these steps in detail, illustrating a comprehensive guide for real-world implementation. The ultimate objective is to develop a software prototype that automatically detects concrete defects through visual analysis to prove the effectiveness of the GENESIS Methodology.

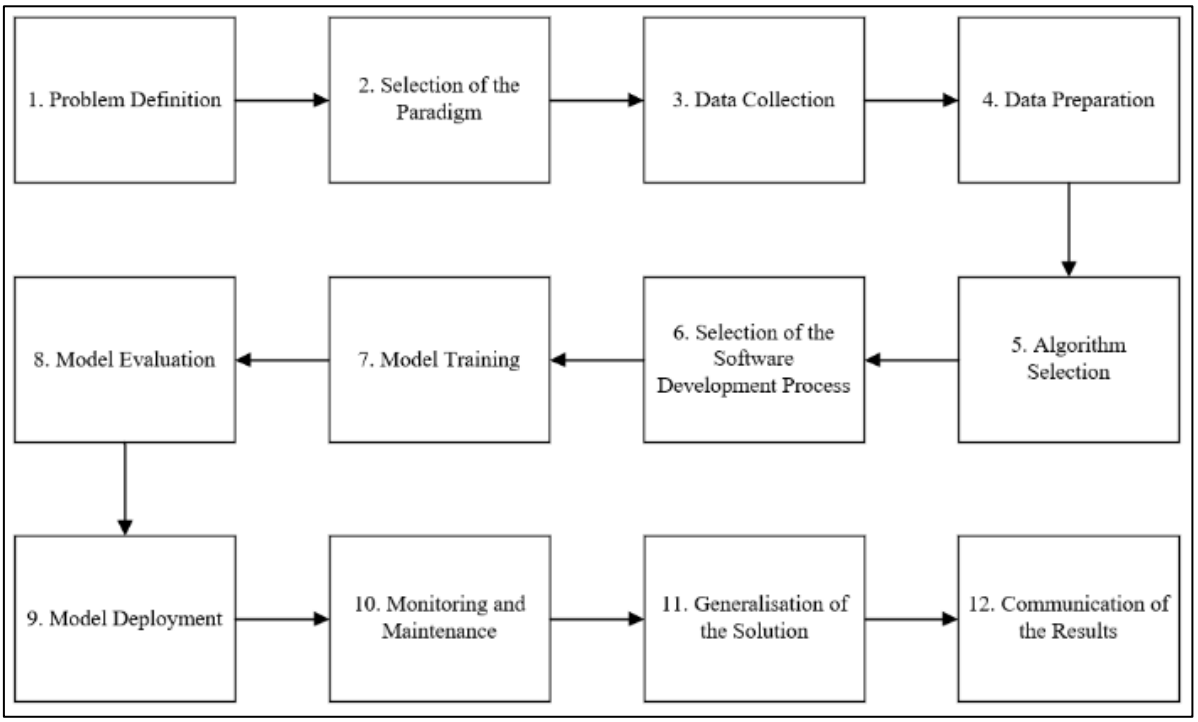

Figure 5: The flow of 12 Stages of the GENESIS Methodology.

1. Problem Definition: It is crucial to understand user needs and requirements and identify the specific problem that needs to be addressed (Oliveira *et al.* 2024; Tian *et al.* 2024). Additionally, it is important to assess whether the proposed solution can only be achieved using AI. The automated detection of defects in concrete requires complex image processing capability. Transferring domain knowledge through insights from domain experts to the AI module and software system is crucial and should be implemented systematically to generate the desired outcome from the system (Perera S *et al.* 2023; Tian *et al.* 2024).
2. Selection of the Paradigm: A selection must be made between data-driven AI and model-driven AI. Data-driven AI aims to build a system capable of identifying the correct answer by analysing numerous question-answer pairs and the received training. Conversely, model-driven AI aims to capture knowledge and facilitate decision-making by employing explicit representation and rules. The decision made

between these two approaches dictates the selection of algorithms and data collection methods. Furthermore, performance can be improved by optimising hyperparameters and utilising ensemble learning techniques such as bagging, stacking, and boosting. Feature engineering is also important to develop new features or to optimise the existing features of the AI model. The software prototype would utilise a model-driven approach with minimum use of training data and the proper deployment of the AI model.

3. Data Collection: AI-embedded software systems heavily rely on data as the output, and the accuracy of the system depends on the selection of the training dataset (Mengiste *et al.* 2022; Sari *et al.* 2023) . The data collection could be performed using multiple sources such as databases, APIs, web scraping, and other means (Saridaki and Haugbølle 2022). The viability of the proposed software system depends on the availability of accurate and relevant datasets (Bang *et al.* 2022). A high-quality image dataset was developed using multiple data sources containing different types of defects in concrete. The data should be selected to minimise the data biases and to incorporate a balanced data set for model training and evaluation. Unbiased data will increase the ability of the AI model to address real-world problems and generate accurate outcomes.

4. Data Preparation: The dataset should be collected and organised and should be pre-processed and cleansed. This is called data augmentation and annotation. This step is critical for AI models to learn and train with high-quality data. The data set is generally divided into two sets: a training set and a test set, with the algorithm being trained on the former and the latter reserved for evaluation. The sampling method applied to the data varies depending on the algorithm's complexity.

5. Algorithm Selection: Choosing suitable AI algorithms and models for problem-solving requires careful consideration (Fitzsimmons *et al.* 2022; Sanni-Anibire *et al.* 2021). It is essential to accurately identify the necessary AI capabilities, as different AI applications will require distinct algorithms (Bhokare *et al.* 2022). The training method should be decided whether to be supervised, unsupervised or reinforced learning (Arun Anoop *et al.* 2024). Further, it is advised to choose an AI model with explainability and transparency in order to enhance fairness and ethical deployment (Perera P *et al.* 2025; Ali *et al.* 2023; Kim S and Park 2023). The software prototype would be focused on the visual computing application and would depend on unsupervised training. The selection of a suitable algorithm was performed through the analysis of 170 research articles that were focused on the development of visual computing applications for built environment applications. The findings of the systematic study are presented below in Figure 6. A systematic literature review was performed to obtain the relevant algorithms for each application.

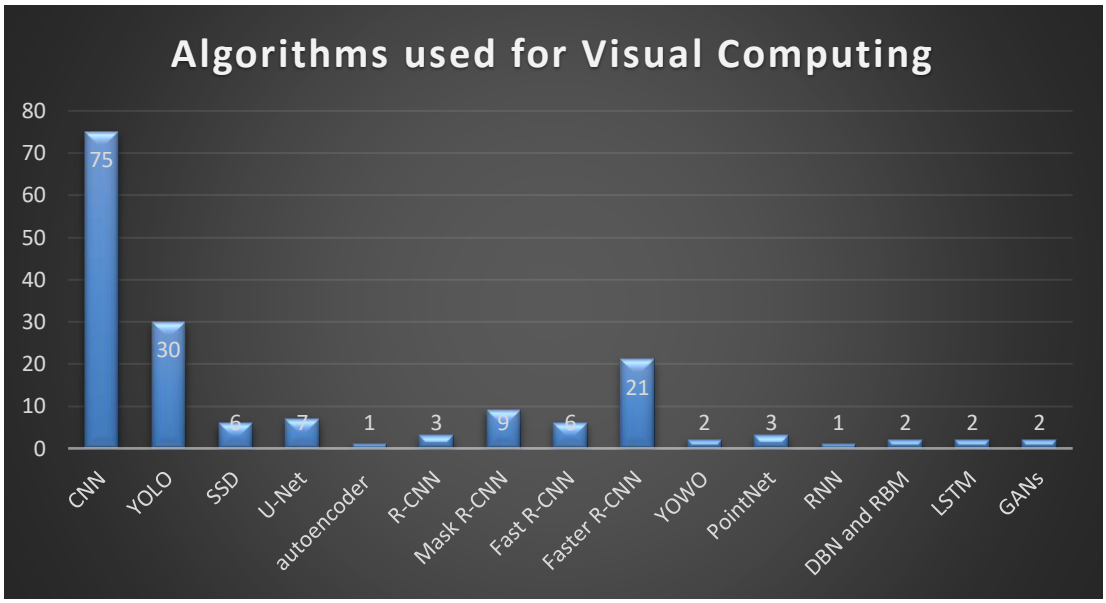

Figure 6: DL algorithms for Visual Computing applications for Construction.

The observed data suggests that visual computing has predominantly relied on the implementation of algorithms derived from Convolutional Neural Networks (CNNs). It is imperative to grasp the intricacies of the CNN algorithm and its significance in the context of visual computing. The software prototype was developed with an AI module containing a CNN, specifically a YOLOv8 deep learning model, based on the specific technical analysis. The single-shot detector DL algorithms, such as YOLO, have provided an optimised approach for object detection to be incorporated into multiple tasks (Bhokare *et al.* 2022; Jiang Z *et al.* 2023). Further, a study conducted by Yu *et al.* (2021) was referred to identify the feature extraction of computer vision applications, since the prototype developed was focused on the same functionality.

One of the key strengths of CNNs is their ability to learn hierarchical features within the input data. As the data flows through the network, the lower-level layers learn small local patterns, while the higher-level layers learn larger patterns synthesised from the features learned in previous layers. This feature makes CNNs well-suited for image analysis and a range of other processing tasks that require the identification and classification of complex patterns within the data. Overall, CNNs offer a powerful and flexible tool for data processing and analysis, which can outperform traditional ANNs in many applications (Al-Shboul *et al.* 2023; Yu *et al.* 2022).

6. Selection of the Software Development Process: The development of an AI-embedded software solution is a highly complex and methodical process. The selection of the software development model is crucial to facilitate the proper requirement capturing and feature engineering, and should consider the paradigm shift from the conventional approach to software development, in which algorithms are manually coded, towards the creation of AI systems that learn from data. Consequently, it is necessary to reassess conventional software development methods and to consider the distinctive requirements posed by these new applications (Giray 2021a). To effectively develop an integrated software system, both the AI-enabled core and the remainder of the software system must be methodically developed, with a focus on comprehensive requirements elicitation, software design, integrated system development, systematic testing, and final deployment. Various software development models have been examined for integrated software system development (Ahmad *et al.* 2023; Goyal *et al.* 2023).

   Martínez-Fernández *et al.* (2022) had emphasised the fact that software engineers developing AI systems generally find it difficult to establish repeatable, systematic software AI development processes. Further approaches were studied for the AI-embedded system development. Goyal *et al.* (2023) had also performed a similar classification.

   1. Waterfall model
   2. V model
   3. Incremental model
   4. RAD model
   5. Agile model
   6. Iterative model
   7. Spiral model

   The above-mentioned software development methodologies were extensively studied to extract the theoretical elements for the justification of using each methodology for AI-embedded software engineering. Ahmad *et al.* (2023) had conducted a comparative analysis of prominent software development approaches and identified limitations associated with iterative software development models. The findings indicate that while iterative methodologies prioritize functionality and user requirements, AI-embedded systems rely entirely on the availability and quality of data. Thus, even with the implementation of advanced and speedy Agile processes, an approach that does not prioritise data and centres around data-centric principles will consistently fall short when confronted with real-world data realities.

   The data-centric nature of AI-embedded systems requires a comprehensive understanding of data acquisition and modelling for the design of the AI-embedded software system. Comparing the V-Model and the Waterfall model, Zhu *et al.* (2019) contended that the V-Model generates inefficient and persistent testing procedures, rendering the evaluation of the AI component of the system challenging. Additionally, it is unsuitable for IT systems with a high degree of complexity, and the software system is produced during the implementation stage, preventing the creation of early prototypes, which is feasible in the

Waterfall model Dagnino *et al.* (2022). Further establishment of this argument could be extracted from the reasons highlighted by Fagarasan *et al.* (2021): a software development project with fixed requirements and scope should use the waterfall model to achieve the lowest risk levels. Since the research project would be delivering the prototype system in a single launch, the waterfall model could be considered the best approach.

The Waterfall model consists of five key steps as illustrated below.

- Requirements analysis
- Design
- Development
- Testing
- Deployment

(Fagarasan *et al.* 2021)

Llinas *et al.* (2021) argued that the waterfall model is chosen for most of the US military AI-based mission-critical software systems development focused on battlefield management, medical, aerospace, and organizational management. Decision makers of large organizations and government agencies were primarily motivated by the level of comfort that following a structured and well-understood process provided. The Department of Defence established a standard, known as "DOD-STD-2167A", which mandated the use of the waterfall methodology as the sanctioned approach for mission-critical software development. After considering all the relevant factors presented above, it was determined that the Waterfall model is the most suitable software development methodology for this research project. The requirement for the software is identified through an extensive process, including domain expert interviews and detailed literature reviews. The UX and the shell of the software will be developed using the Waterfall model. The AI model to be embedded in the software system is selected and developed based on the requirement, yet will be refined and retrained during the next stages where the focus is on the AI model training, evaluation and deployment. This hybrid strategy enables the iterative improvement of the AI component, ensuring the flexibility of the methodology being responsive to real-world data dynamics despite the disciplined framework provided by the Waterfall model to capture the complex requirements.

7. Model Training: After the algorithm selection process is complete, the subsequent step involves training the model on the data that has been collected. This process is initiated with the establishment and selection of suitable hyperparameters, optimisation of the model's performance, and validation of the results through testing with a separate set of data. In a model-driven approach, it is possible to consider ensemble learning, which combines the outputs of multiple models to improve the overall accuracy of predictions and deep reinforcement learning (Horneman *et al.* 2020). Further, Nascimento E *et al.* (2020) had proposed re-enforced learning as a model training mechanism for Deep Learning (DL) models. Ding *et al.* (2023) had specified seven mechanisms to train a DL model. The proposed software prototype is trained using the unsupervised training method since the application requires the model building within the AI algorithm.
8. Model Evaluation: Once the training process is completed, the model evaluation is performed to assess the performance of the learnt model. The evaluation comprises the calculation of multiple mathematical metrics such as accuracy, precision, and recall. A distinct data set other than the training dataset is utilised to evaluate the AI model. The testing dataset is used to evaluate the model's ability to generalise to new data and to estimate the model's performance on unseen data. Mulindwa and Du (2023) had emphasised the importance of considering the activation functions in model evaluation. Further, Yang *et al.* (2023) highlighted the possible backpropagation to enhance the accuracy of the AI model.

   The scientific validation of a DL model is established through the measurement of precision (Ozkaya Ipek 2023). The determination of the accuracy and validity of the proposed prototype would be based on the statistical scientific method employed. In the statistical analysis of binary classification, the F-score or F-measure is utilized to determine the accuracy of a test. The F-score is derived from the test's precision

and recall, where precision represents the number of true positive results divided by the total number of positive results, including those not identified correctly, and recall represents the number of true positive results divided by the total number of all samples that should have been identified as positive. The mean average precision (mAP) measures the accuracy of the predictions. The mean Average Precision (mAP) score is calculated by taking the mean average precision over all possibilities (Ramzi *et al.* 2022). Computer vision applications utilise the same principle to check the accuracy and depend on the object detection capability of the prototype (Gezici and Tarhan 2022; Huang Q. and Hao 2020). Special testing methods should be utilised for testing the AI-embedded systems, and it is the most common area of Software Engineering focused on specialised AI-embedded software development (Giray 2021a).

9. Model Deployment: The deployment of the AI model by integrating it into the complete software system. The other modules of the software system were developed systematically using software development tools such as Unified Modelling Language (UML) and Prototyping. The AI model was embedded into a separate AI module, and the other software modules were interconnected with an API. Additionally, standard software testing processes would be employed to ensure the reliability and efficiency of the deployed model. The integration process would be carried out while adhering to industry-standard software development practices to guarantee the quality of the final product.

10. Monitoring and Maintenance: Once the deployment of the AI module and the other software modules are completed and tested, a comprehensive evaluation should be started. Monitoring the AI model once usage starts is essential since the accuracy of the AI model might change due to its usage. The monitoring process involves evaluating the performance of the model by analysing its outputs and detecting any anomalies. Additionally, new data needs to be collected to retrain the model and improve its performance over time. This process requires continuous data acquisition and management to ensure the model remains up-to-date and accurate. Regular evaluation of the AI model using the key evaluation metrics with iterative AI model refinement (e.g., accuracy, precision, recall) will be performed to enhance the adaptability of the system concerning evolving real-world conditions. MLOps-inspired techniques such as automated performance tracking, drift detection, and model retraining would be integrated to ensure the applicability and effectiveness of the outcomes of the software system. Periodical reviews from the domain experts related to the system's overall outcomes are essential to maintain the applicability of the software system. AI Model versioning and continuous integration and deployment (CI/CD) will be performed as per guidelines defined through the MLOps methodology. Furthermore, updating the model with new features or bug fixes is necessary to improve the overall performance of the system. Regular maintenance of the model would involve debugging, upgrading, and refining the model to ensure that it continues to meet the desired standards and specifications. This structured approach guarantees that the AI system remains robust, unbiased, and effective throughout its lifecycle.

The developed software prototype strictly followed the above-mentioned steps and the methods adhering to the specific steps mentioned in each stage. The next section of the study illustrates the software prototype development and the illustrations.

4. PROOF OF METHODOLOGY: DEVELOPMENT OF THE AI-EMBEDDED AUTOMATED CONCRETE DEFECT DETECTION SOFTWARE PROTOTYPE

The developed software prototype is the practical implementation of the innovative GENESIS Methodology, detailed and presented in this study. Designed specifically for concrete defect detection, this prototype integrates multiple modules to perform a variety of tasks. The development of this prototype adhered strictly to the GENESIS Methodology defined in this research, following the Waterfall model as justified in the methodology section. Each phase of development was executed with the proper process and documentation. Key requirements identified included the capability to detect multiple types of concrete defects, the ability to record and manage image metadata such as location and building element details, integration with a database for efficient data management, and a user-friendly interface for image upload and result visualisation.

The AI Module, housing the YOLOv8-based defect detection model, forms the technological core of the system. This is complemented by the Data Management Module, which handles image metadata and database interactions. The Image Processing Module manages the pre-processing and post-processing of images, optimising them for

analysis and result presentation. Finally, the User Interface Module provides a seamless experience for users interacting with the system and interpreting results.

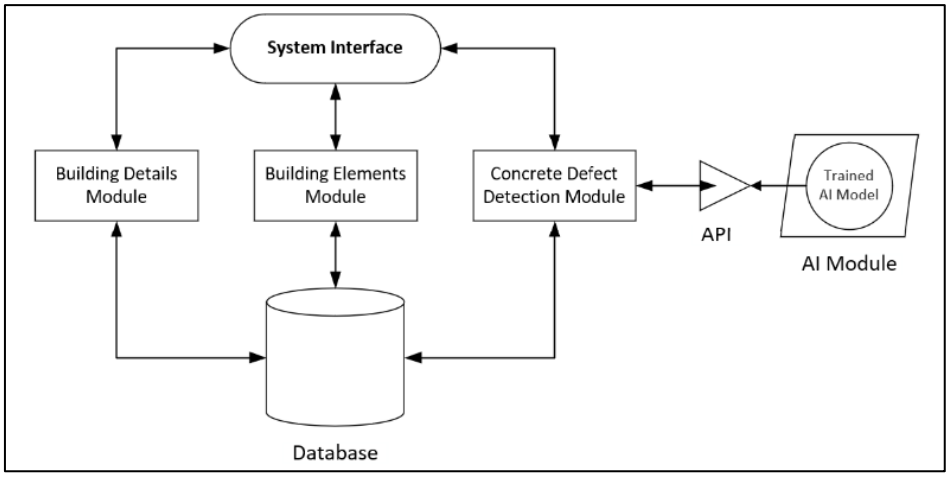

Figure 7: System Architecture with regular software modules and the AI module.

The system architecture is illustrated in Figure 7 above. The modules interact with the system interface for data exchange. The modules utilise a single database across the system. The 'AI module' includes the trained YOLOv8 embedded. The image data flows from the system interface to the 'Defect Detection Module', and it passes the refined input image to the 'AI module'. The YOLOv8 model extracts the features of the input image and passes the mask and the detection box through the API. The 'Defect Detection Module' recreates the input image, and if a defect is detected, it creates an imposed box with the defect category.

The development of the software system was performed using Python as the primary programming language. Through research, six distinct concrete defect types were identified as the focus of the detection system, specifically: cracks, spalling, efflorescence, exposed reinforcement, honeycombing, and staining. This classification is implemented in the system's Python code, allowing for clear categorisation and reporting of detected defects.

```
class_colors = {
    "Exposed Reinforcement": (0, 255, 0),  # Green
    "Ruststrain": (0, 0, 255),  # Red
    "Scaling": (255, 0, 0),  # Blue
    "Spalling/Honeycombing": (191, 0, 225), # Purple
    "Concrete Crack": (207, 54, 133), #  Pink
    "Concrete Efflorescence": (255, 242, 0), # Yellow
```

Figure 8: Six defect types defined in the Python programming code.

A comprehensive dataset was compiled from online image databases and carefully pre-processed to provide a diverse and representative sample for training and testing the AI model. The dataset consisted of 1200 labelled images sourced from repositories on GitHub and supplemented with real-world images captured from construction sites. The dataset was preprocessed using image augmentation techniques to extract the optimum feature from the model. The image dataset was divided into 80-20 proportions in order to create the training dataset and the testing dataset. Training the model was an iterative process based on the methodology defined and with reference to MLOps methodologies. The training of the AI model was performed in multiple stages. First, it was trained with a batch size of 50 and 5 epochs to check the effectiveness of the image dataset by calculating the mAP (Mean Average Precision), F1-score, recall, and precision metrics. Once it was monitored for positive mAP, it was trained for two iterations with 20 epochs and 30 epochs. All models were saved separately as illustrated in Figure 9. Finally, the model was trained with 100 epochs and was established as the final version. The multiple models had their parameters fine-tuned for optimal recognition of the six identified defect types. The trained model was then embedded into a dedicated AI module, ready for integration with the complete system

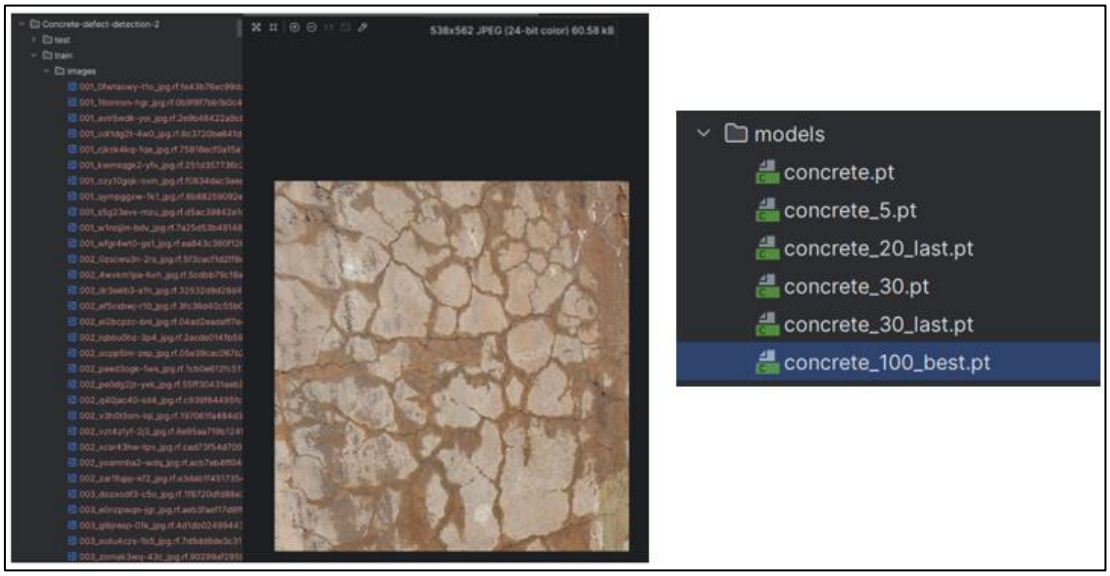

Figure 9: Illustration of the Image Dataset and the saved YOLOv8 models trained with multiple epochs.

Parallel to the AI development, the other modules were developed and an API was developed to facilitate communication between these modules, ensuring system cohesion. This modular approach, guided by the principles outlined in the methodology, allowed for flexible development and easier testing of individual components. Initial evaluation of the AI model revealed an impressive accuracy of 91.2% on the test dataset, validating the effectiveness of the YOLO algorithm and training process. Each module underwent individual testing to ensure proper functionality before proceeding to integration testing. The fully integrated system was then subjected to comprehensive testing, verifying the interaction between all components.

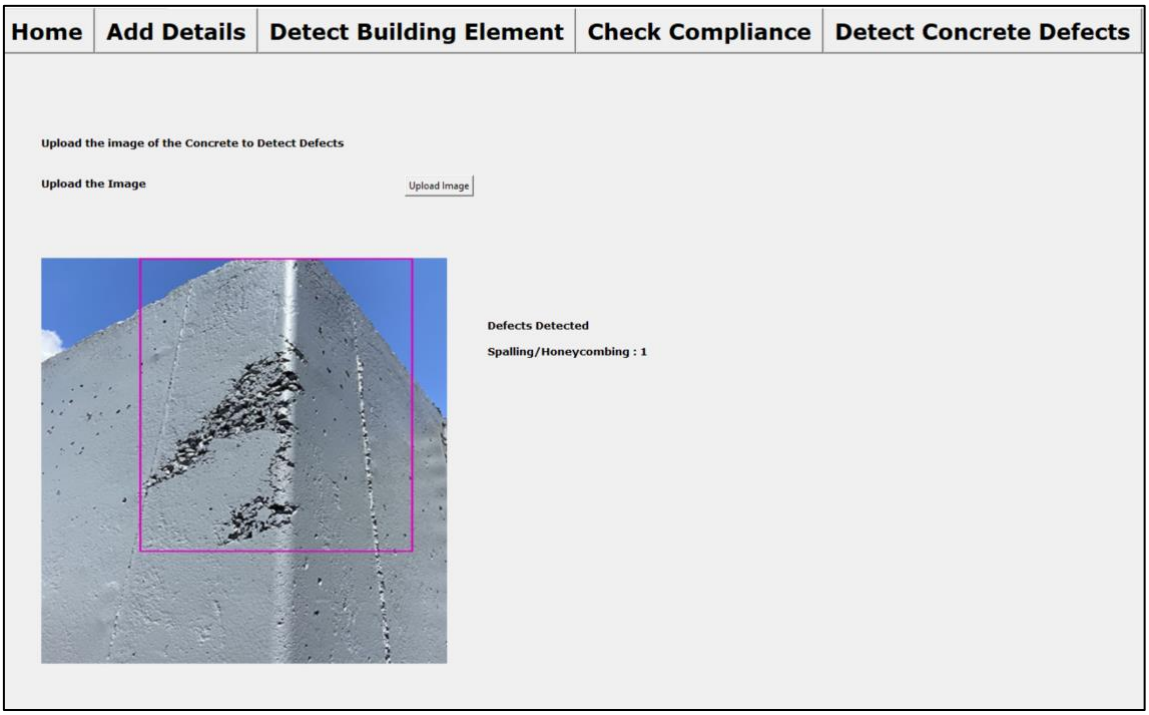

Figure 10: Final evaluation: Detection of Honeycombing in Concrete.

The user interface of the prototype has been designed with a simplified user experience. It facilitates the easy upload of concrete images, provides clear visualisation of detected defects, and offers access to detailed defect information and associated metadata. The database used is MySQL for the regular modules, to save data related to user management, building details and building elements. This ensures that the complex analytical capabilities of the system are presented in a simple user interaction. The following figures demonstrate multiple concrete defects being identified in the system. Figure 9 illustrates the detection of Honeycombing in Concrete, while Figure 10 illustrates concrete cracks. Figure 11 illustrates the detection of multiple defects in concrete.

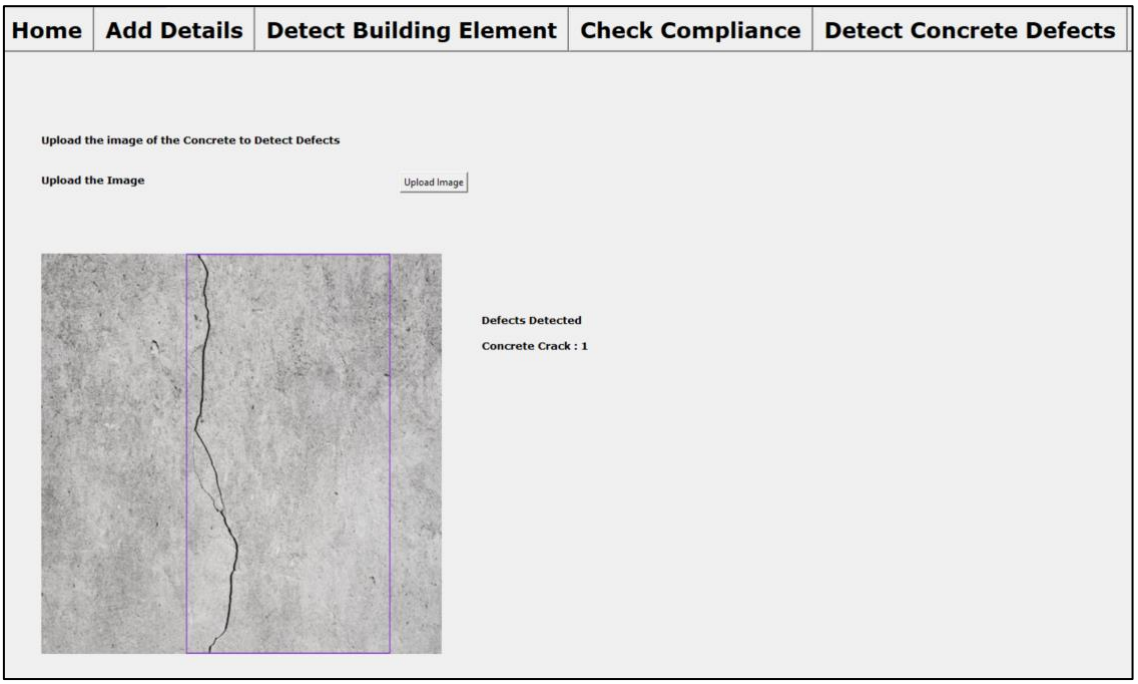

Figure 11: Final evaluation: Detection of Cracks in Concrete.

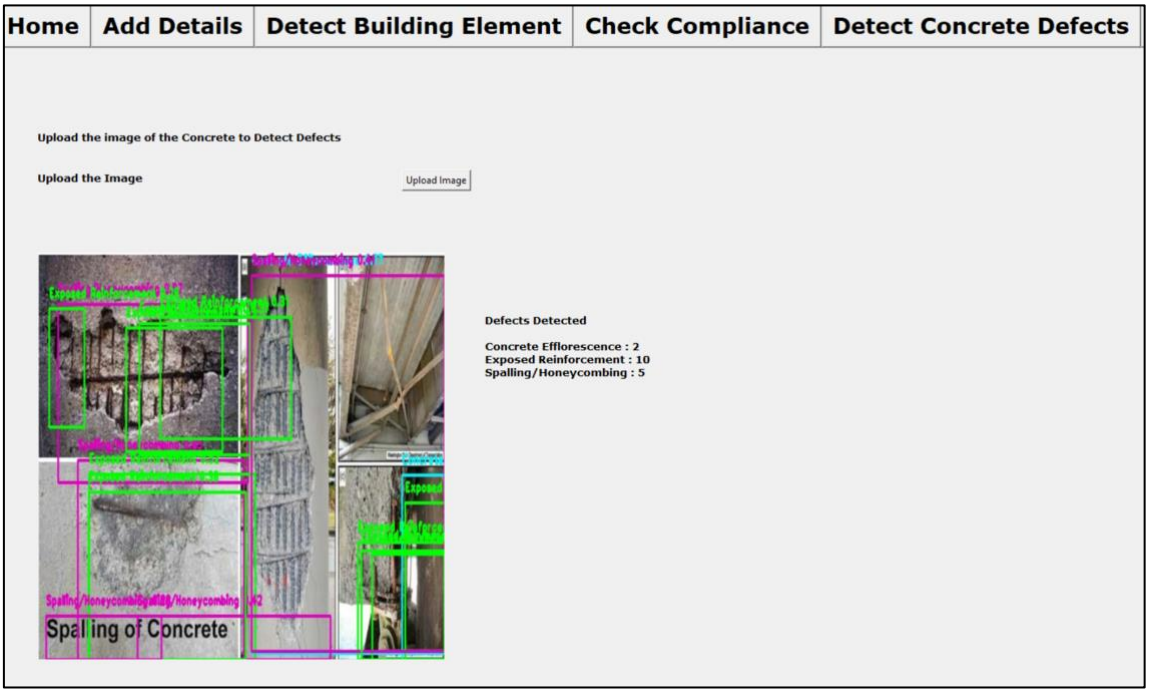

Figure 12: Final evaluation: Detection of multiple defects in concrete.

The successful integration and performance of this prototype serve as proof of concept for the innovative and comprehensive methodology developed for AI-embedded software systems. It demonstrates the effective implementation of the Waterfall model adapted for AI development, illustrates the integration of AI capabilities with traditional software modules, and achieves a high level of accuracy in concrete defect detection. Most importantly, it illustrates the practical applicability of the proposed methodology to real-world problems in the built environment, validating its potential for developing robust, AI-embedded software solutions across various domains.

5. CONCLUSION AND FURTHER RESEARCH

This research has presented an innovative software development methodology, the GENESIS Methodology, for developing AI-embedded software systems, with a specific focus on applications within the built environment. The concept was proven with the successful development and implementation of an automated concrete defect detection prototype. Further, it was able to address the unique challenges inherent in developing AI-embedded systems for visual computing tasks. The GENESIS Methodology integrates elements from established software development frameworks and AI-specific features and resolves the unique challenges posed by AI-embedded systems. By adopting a Waterfall model approach, enhanced with AI-specific stages such as data preparation, algorithm selection, and model training, the GENESIS Methodology provides a structured yet flexible framework for developing complex AI-embedded software.

One of the key strengths of the GENESIS Methodology lies in its comprehensive nature. All stages of software development are considered in the study, from initial problem definition and paradigm selection through to deployment and ongoing maintenance. This universal approach ensures that critical aspects specific to AI systems, such as data quality and model performance, are given due consideration throughout the development process. The GENESIS Methodology's integrated iterative evaluation and refinement of the AI model, coupled with conventional software engineering practices, contributes to the development of more robust and reliable AI-embedded systems.

The concrete defect detection prototype, which was developed as the proof of concept, validated the GENESIS Methodology by achieving a 91.2% accuracy in identifying six types of concrete defects, demonstrating the accuracy of this approach in real-world applications. Furthermore, the GENESIS Methodology combines rigorous requirement analysis, iterative model refinement, and comprehensive lifecycle management through the synthesis of multiple AI-embedded system development methodologies. It was able to achieve high accuracy once deployed as demonstrated through the software prototype, and also streamlined development compared to conventional methodologies. While this study focused on the built environment, the GENESIS Methodology has potential applications across various domains. The GENESIS methodology can be expanded to other domains with minimum modifications by adjusting domain-specific factors, such as data sources, model selection, and performance evaluation criteria. Further research could be focused on its adaptability to other sectors, each presenting unique challenges and opportunities for AI-embedded software systems. Further research into effective models for collaboration between software engineers, data scientists, and domain experts in the built environment could enhance the practical application of the GENESIS Methodology.

Despite the above-mentioned contributions to the study, certain limitations in the developed prototype should be acknowledged. The scope of the application is focused on a single use case and only on six predefined defect types, and it aims to illustrate the applicability to broader visual computing tasks. Data dependencies and model generalisation were not considered for the prototype since the objective is to establish the applicability of the GENESIS methodology for AI-embedded software development. The YOLOv8 model utilised in the prototype software lacks advanced explainability features, and integration of Explainable AI (XAI) techniques in future iterations will enhance transparency and allow stakeholders to better interpret the model's decision-making processes.

In conclusion, this research has made a significant contribution to the field of AI-embedded software engineering by proposing and validating the comprehensive GENESIS Methodology. As AI continues to evolve and deploy various aspects of our technological landscape, methodologies such as GENESIS will play a crucial role in ensuring the development of robust, efficient, and ethically sound AI-embedded software systems.

ACKNOWLEDGEMENT

We would like to express our gratitude to Commnia Pty Ltd for co-funding the PhD research project, which enabled the execution of this research and the publication of its outcomes.

REFERENCES

Ahmad K., Abdelrazek M., Arora C., Bano M. and Grundy J. (2023). Requirements engineering for artificial intelligence systems: A systematic mapping study, Information and Software Technology, vol. 158, p. 107176, DOI: 10.1016/j.infsof.2023.107176.

Akbar M.A., Khan A.A., Mahmood S. and Rafi S. (2024). Quantum Software Engineering: A New Genre of Computing, Proceedings of the 1st ACM International Workshop on Quantum Software Engineering: The Next Evolution. Association for Computing Machinery, DOI: 10.1145/3663531.3664750.

Al-Shboul A., Gharibeh M., Najadat H., Ali M. and El-Heis M. (2023). Overview of convolutional neural networks architectures for brain tumour segmentation, International Journal of Electrical and Computer Engineering, vol. 13, no. 4, pp. 4594-604, DOI: 10.11591/ijece.v13i4.pp4594-4604.

Alevizos V., Georgousis I., Simasiku A., Messinis A., Karypidou S. and Malliarou D. (2024). Evaluating the Inclusiveness of Artificial Intelligence Software in Enhancing Project Management Efficiency - A review and examples of quantitative measurement methods, DOI: 10.1109/ACDSA59508.2024.10467463.

Ali S., Abuhmed T., El-Sappagh S., Muhammad K., Alonso-Moral J.M., Confalonieri R. et al. (2023). Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence, Information Fusion, vol. 99, p. 101805, DOI: 10.1016/j.inffus.2023.101805.

Amershi S., Begel A., Bird C., DeLine R., Gall H., Kamar E. et al. (2019). Software Engineering for Machine Learning: A Case Study, pp. 291-300, DOI: 10.1109/ICSE-SEIP.2019.00042.

Arun Anoop M., Karthikeyan P. and Poonkuntran S. 2024. 'Unsupervised/Supervised Feature Extraction and Feature Selection for Multimedia Data (Feature Extraction with Feature Selection for Image Forgery Detection)', in Supervised and Unsupervised Data Engineering for Multimedia Data, pp. 27-61.

Bang S., Aarvold M.O., Hartvig W.J., Olsson N.O.E. and Rauzy A. (2022). Application of machine learning to limited datasets: prediction of project success, Journal of Information Technology in Construction, vol. 27, pp. 732-55, DOI: 10.36680/j.itcon.2022.036.

Barenkamp M., Rebstadt J. and Thomas O. (2020). Applications of AI in classical software engineering, AI Perspectives, vol. 2, no. 1, p. 1, DOI: 10.1186/s42467-020-00005-4.

Batarseh F.A., Bui J., Chong D., Eisenberg S., Gendron J., Guerrero J.M. et al. 2020. 'Data Democracy At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering', in FA Batarseh and R Yang (eds), Data Democracy, Academic Press, p. vii.

Beneder R. and Schmitt P. (2024). Introduction of a Remote Lab for Indoor Object Position Control Based on Computer Vision Sensors and AI-Enabled Embedded Systems, pp. 1-5, DOI: 10.1109/EDUCON60312.2024.10578864.

Bhokare S., Goyal L., Ren R. and Zhang J. (2022). Smart construction scheduling monitoring using yolov3-based activity detection and classification, Journal of Information Technology in Construction, vol. 27, pp. 240-52.

Bier H., Hidding A., Khademi S., van Engelenbrug C., Alavi H. and Zhong S. (2024). Advancing Applications for Artificial-Intelligence-Supported Ambient Control in the Built Environment, Technology|Architecture + Design, vol. 8, no. 1, pp. 155-64, DOI: 10.1080/24751448.2024.2322927.

Carstensen A.-K. and Bernhard J. (2019). Design science research – a powerful tool for improving methods in engineering education research, European Journal of Engineering Education, vol. 44, no. 1-2, pp. 85-102, DOI: 10.1080/03043797.2018.1498459.

Clement T., Kemmerzell N., Abdelaal M. and Amberg M. (2023). XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process, Machine Learning and Knowledge Extraction, vol. 5, no. 1, pp. 78-108.

Conboy K., Ågerfalk P., Crowston K., Lundström J., Jarvenpaa S., Ram S. et al. (2021). Artificial Intelligence in Information Systems: State of the Art and Research Roadmap, Communications of the Association for Information Systems, vol. 50, DOI: 10.17705/1CAIS.05017.

Dagnino A., Kolomycki M. and Kucheria A. (2022). MAP: Design, Development, Deployment, and Maintenance of Industrie 4.0 AI Applications, pp. 108-13, DOI: 10.1109/BigDataService55688.2022.00024.

Dakkak A., Bosch J. and Holmstrom Olsson H. (2024). Towards AIOps enabled services in continuously evolving software-intensive embedded systems, Journal of Software: Evolution and Process, vol. 36, no. 5, p. e2592, DOI: 10.1002/smr.2592.

Ding Y., Liu C., Zhu H. and Chen Q. (2023). A supervised data augmentation strategy based on random combinations of key features, Information Sciences, vol. 632, pp. 678-97, DOI: 10.1016/j.ins.2023.03.038.

Domingos P. 2015. The master algorithm: How the quest for the ultimate learning machine will remake our world, Basic Books.

Doukari O., Greenwood D., Rogage K. and Kassem M. (2022). Object-centred automated compliance checking: a novel, bottom-up approach, Journal of Information Technology in Construction, vol. 27, pp. 335-62, DOI: 10.36680/j.itcon.2022.017.

Drechsler A. and Hevner A. 2016. A Four-Cycle Model of IS Design Science Research: Capturing the Dynamic Nature of IS Artifact Design.

Dresch A., Lacerda D. and Antunes Júnior J.A.V. 2014. Design Science Research: A Method for Science and Technology Advancement.

Ekanayake B., Wong J.K.-W., Fini A.A.F. and Smith P. (2021). Computer vision-based interior construction progress monitoring: A literature review and future research directions, Automation in Construction, vol. 127, p. 103705, DOI: doi.org/10.1016/j.autcon.2021.103705.

Eliwa H.K., Jelodar M.B., Poshdar M. and Yi W. (2023). Information and communication technology applications in construction organizations: a scientometric review, Journal of Information Technology in Construction, vol. 28, pp. 286-305, DOI: 10.36680/J.ITCON.2023.014.

Engström E., Storey M.-A., Runeson P., Höst M. and Baldassarre M.T. (2020). How software engineering research aligns with design science: a review, Empirical Software Engineering, vol. 25, no. 4, pp. 2630-60, DOI: 10.1007/s10664-020-09818-7.

Fagarasan C., Popa O., Pisla A. and Cristea C. (2021). Agile, waterfall and iterative approach in information technology projects, IOP Conference Series: Materials Science and Engineering, vol. 1169, no. 1, p. 012025, DOI: 10.1088/1757-899X/1169/1/012025.

Figueiredo A.C., Pereira R. and da Silva M.Â. 2025. 'Exploring the Integration of Artificial Intelligence and DevOps for Agile Product Development', in R Pereira, I Bianchi andA Rocha (eds), Digital Technologies and Transformation in Business, Industry and Organizations: Volume 3, Springer Nature Switzerland, Cham, pp. 27-39.

Fitzsimmons J.P., Lu R., Hong Y. and Brilakis I. (2022). Construction schedule risk analysis - a hybrid machine learning approach, Journal of Information Technology in Construction, vol. 27, pp. 70-93, DOI: 10.36680/j.itcon.2022.004.

Gestão, Produção G., Goecks L., Souza M., Librelato T. and Trento L. (2021). Design Science Research in practice: review of applications in Industrial Engineering, Gestão & Produção, vol. 28, p. 5811, DOI: 10.1590/1806-9649-2021v28e5811.

Gezici B. and Tarhan A.K. (2022). Systematic literature review on software quality for AI-based software, Empirical Software Engineering, vol. 27, no. 3, p. 66, DOI: 10.1007/s10664-021-10105-2.

Giray G. (2021a). A software engineering perspective on engineering machine learning systems: State of the art and challenges, Journal of Systems and Software, vol. 180, p. 111031, DOI: 10.1016/j.jss.2021.111031.

Giray G. 2021b. A Software Engineering Perspective on Engineering Machine Learning Systems: State of the Art and Challenges.

Goyal S., Gupta A. and Jha H. (2022). Current Trends in Methodology for Software Development Process, vol. 493, pp. 621-9, DOI: 10.1007/978-981-19-4990-6_58.

Gregor S. and Hevner A.R. (2013). Positioning and Presenting Design Science Research for Maximum Impact, MIS Quarterly, vol. 37, no. 2, pp. 337-55, viewed 2023/04/19/.

Harichandran A., Raphael B. and Mukherjee A. (2023). Relevance of deep sequence models for recognising automated construction activities: a case study on a low-rise construction system, Journal of Information Technology in Construction, vol. 28, pp. 458-81, DOI: 10.36680/j.itcon.2023.023.

Heroux M.A. (2023). Research Software Science: Expanding the Impact of Research Software Engineering, Computing in Science & Engineering, pp. 1-7, DOI: 10.1109/MCSE.2023.3260475.

Hevner A., R A., March S., T S., Park, Park J. et al. (2004). Design Science in Information Systems Research, Management Information Systems Quarterly, vol. 28, p. 75.

Hewavitharana S., Perera A., Perera S., Perera P. and Nanayakkara S. (2025). Framework for systematic adoption of ERP systems in the Sri Lankan construction industry, Built Environment Project and Asset Management, vol. ahead-of-print, no. ahead-of-print, viewed 2025/01/28, DOI: 10.1108/BEPAM-12-2023-0226.

Heyn H.-M., Knauss E., Pir A., Erikssonz O., Linder J., Subbiah P. et al. 2021. Requirement Engineering Challenges for AI-intense Systems Development.

Horkoff J. (2019). Non-Functional Requirements for Machine Learning: Challenges and New Directions, pp. 386-91, DOI: 10.1109/RE.2019.00050.

Horneman A., Mellinger A.O. and Ozkaya I. AI Engineering: 11 Foundational Practices.

Hossain Faruk M.J., Pournaghshband H. and Shahriar H. AI-Oriented Software Engineering (AIOSE): Challenges, Opportunities, and New Directions, vol. 576 LNNS, pp. 3-19, DOI: 10.1007/978-3-031-20322-0_1.

Hu B.C., Salay R., Czarnecki K., Rahimi M., Selim G. and Chechik M. Towards Requirements Specification for Machine-learned Perception Based on Human Performance, pp. 48-51, DOI: 10.1109/AIRE51212.2020.00014.

Huang Q. and Hao K. (2020). Development of cnn-based visual recognition air conditioner for smart buildings, Journal of Information Technology in Construction, vol. 25, pp. 361-73, DOI: 10.36680/j.itcon.2020.021.

Huang Q., Rodriguez K., Whetstone N. and Habel S. (2019). Rapid Internet of Things (IoT) prototype for accurate people counting towards energy efficient buildings, Journal of Information Technology in Construction, vol. 24, pp. 1-13, DOI: 10.36680/j.itcon.2019.001.

Islam Z.U. Software engineering methods for responsible artificial intelligence, vol. 3, pp. 1802-3.

Janiesch C., Zschech P. and Heinrich K. (2021). Machine learning and deep learning, Electronic Markets, vol. 31, no. 3, pp. 685-95, DOI: 10.1007/s12525-021-00475-2.

Jiang A., Mo Y. and Kalasapudi V.S. (2022). Status quo and challenges and future development of fire emergency evacuation research and application in built environment, Journal of Information Technology in Construction, vol. 27, pp. 781-801, DOI: 10.36680/j.itcon.2022.038.

Jiang Z., Messner J.I. and Matts E. (2023). Computer vision applications in construction and asset management phases: a literature review, Journal of Information Technology in Construction, vol. 28, pp. 176-99, DOI: 10.36680/J.ITCON.2023.009.

John M.M., Olsson H.H. and Bosch J. (Year) of Conference. Towards MLOps: A Framework and Maturity Model, pp. 1-8, DOI: 10.1109/SEAA53835.2021.00050.

Kakatkar C., Bilgram V. and Füller J. (2020). Innovation analytics: Leveraging artificial intelligence in the innovation process, Business Horizons, vol. 63, no. 2, pp. 171-81, DOI: 10.1016/j.bushor.2019.10.006.

Khomh F., Adams B., Cheng J., Fokaefs M. and Antoniol G. (2018). Software Engineering for Machine-Learning Applications: The Road Ahead, IEEE Software, vol. 35, no. 5, pp. 81-4, DOI: 10.1109/MS.2018.3571224.

Kim M., Zimmermann T., DeLine R. and Begel A. (2018). Data Scientists in Software Teams: State of the Art and Challenges, IEEE Transactions on Software Engineering, vol. 44, no. 11, pp. 1024-38, DOI: 10.1109/TSE.2017.2754374.

Kim S. and Park C.S. (2023). Quantification of occupant response to influencing factors of window adjustment behavior using explainable AI, Energy and Buildings, vol. 296, p. 113349, DOI: 10.1016/j.enbuild.2023.113349.

Kreuzberger D., Kühl N. and Hirschl S. (2023). Machine Learning Operations (MLOps): Overview, Definition, and Architecture, IEEE Access, vol. 11, pp. 31866-79, DOI: 10.1109/ACCESS.2023.3262138.

Kulkarni R.H. and Padmanabham P. (2017). Integration of artificial intelligence activities in software development processes and measuring effectiveness of integration, IET Software, vol. 11, no. 1, pp. 18-26, DOI: 10.1049/iet-sen.2016.0095.

Kurrek P., Zoghlami F., Jocas M., Stoelen M. and Vahid S. (2020). Q-Model: An Artificial Intelligence Based Methodology for the Development of Autonomous Robots, Journal of Computing and Information Science in Engineering, vol. 20, pp. 1-16, DOI: 10.1115/1.4046992.

Linares-Vásquez A.M.a.C.E.-V.a.M. (2024). The Rise and Fall(?) of Software Engineering, <https://arxiv.org/abs/2406.10141>.

Llinas J., Fouad H. and Mittu R. 2021. 'Systems Engineering for Artificial Intelligence-based Systems: A Review in Time', in WF Lawless, R Mittu, DA Sofge, T Shortell andTA McDermott (eds), Systems Engineering and Artificial Intelligence, Springer International Publishing, Cham, pp. 93-113.

Lwakatare L.E., Crnkovic I. and Bosch J. DevOps for AI – Challenges in Development of AI-enabled Applications, pp. 1-6, DOI: 10.23919/SoftCOM50211.2020.9238323.

Martin J., Cantero D., González M., Cabrera A., Larrañaga M., Maltezos E. et al. (2022). Embedded Vision Intelligence for the Safety of Smart Cities, Journal of Imaging, vol. 8, no. 12, DOI: 10.3390/jimaging8120326.

Martínez-Fernández S., Bogner J., Franch X., Oriol M., Siebert J., Trendowicz A. et al. (2022). Software Engineering for AI-Based Systems: A Survey, ACM Trans. Softw. Eng. Methodol., vol. 31, no. 2, p. Article 37e, DOI: 10.1145/3487043.

McDermott T.A., Blackburn M.R. and Beling P.A. 2021. 'Artificial Intelligence and Future of Systems Engineering', in WF Lawless, R Mittu, DA Sofge, T Shortell andTA McDermott (eds), Systems Engineering and Artificial Intelligence, Springer International Publishing, Cham, pp. 47-59.

Mengiste E., de Soto B.G. and Hartmann T. (2022). Recognition of the condition of construction materials using small datasets and handcrafted features, Journal of Information Technology in Construction, vol. 27, pp. 951-71, DOI: 10.36680/j.itcon.2022.046.

Mulindwa D.B. and Du S. (2023). An n-Sigmoid Activation Function to Improve the Squeeze-and-Excitation for 2D and 3D Deep Networks, Electronics (Switzerland), vol. 12, no. 4, DOI: 10.3390/electronics12040911.

Munappy A., Bosch J., Olsson H.H., Arpteg A. and Brinne B. Data Management Challenges for Deep Learning, pp. 140-7, DOI: 10.1109/SEAA.2019.00030.

Nascimento E., Nguyen Duc A., Sundbø I. and Conte T. 2020. Software engineering for artificial intelligence and machine learning software: A systematic literature review.

Nascimento E.d.S., Ahmed I., Oliveira E., Palheta M.P., Steinmacher I. and Conte T. Understanding Development Process of Machine Learning Systems: Challenges and Solutions, pp. 1-6, DOI: 10.1109/ESEM.2019.8870157.

Nilsson J., Javed S., Albertsson K., Delsing J., Liwicki M. and Sandin F. (2024). AI Concepts for System of Systems Dynamic Interoperability, Sensors, vol. 24, no. 9, p. 2921.

Oliveira A., Granja J., Bolpagni M., Motamedi A. and Azenha M. (2024). Development of standard-based information requirements for the facility management of a canteen, Journal of Information Technology in Construction, vol. 29, pp. 281-307, DOI: 10.36680/j.itcon.2024.014.

Ozkaya I. (2020). What Is Really Different in Engineering AI-Enabled Systems?, IEEE Software, vol. 37, no. 4, pp. 3-6, DOI: 10.1109/MS.2020.2993662.

Ozkaya I. (2023). The Next Frontier in Software Development: AI-Augmented Software Development Processes. vol. 40, 04, IEEE Computer Society, pp. 4-9, <https://doi.ieeecomputersociety.org/10.1109/MS.2023.3278056>.

Paskaleva G., Mazak-Huemer A., Villeneuve M. and Waldhart J. (2023). Automated translation from domain knowledge to software model: excel2uml in the tunneling domain, Journal of Information Technology in Construction, vol. 28, pp. 360-84, DOI: 10.36680/j.itcon.2023.019.

Peffers K., Tuunanen T., Rothenberger M. and Chatterjee S. (2007). A design science research methodology for information systems research, Journal of Management Information Systems, vol. 24, pp. 45-77.

Perera P., Perera S., Xiaohua J., Rashidi M., Yazbek G., Yazbek A. et al. (2023). Application of Visual Computing and Deep Learning in the Construction Industry, *Proceedings of the Australasian Universities Building Education Association (AUBEA) 2023 Conference*, New Zealand.

Perera P., Perera S., Xiaohua J., Rashidi M., Yazbek G., Yazbek A. et al. (2025). Impact of Explainable Artificial Intelligence for Sustainable Built Environment, *Proceedings of the CIB World Building Congress (CIBWBC) 2025 Conference*, USA.

Perera P., Perera S., Xiaohua J., Rashidi M., Yazbek G., Yazbek A. et al. (2024). The Integration of Geospatial Artificial Intelligence for Enhancing Built Environment Sustainability, *Proceedings of the Australasian Universities Building Education Association (AUBEA) 2024 Conference,* Australia.

Perera S., Jin X., Samaratunga M. and Gunasekara K. (2023). Drivers and barriers to digitalisation: a cross-analysis of the views of designers and builders in the construction industry, Journal of Information Technology in Construction, vol. 28, pp. 87-106, DOI: 10.36680/j.itcon.2023.005.

Perrier N., Bled A., Bourgault M., Cousin N., Danjou C., Pellerin R. et al. (2020). Construction 4.0: A survey of research trends, Journal of Information Technology in Construction, vol. 25, pp. 416-37, DOI: 10.36680/J.ITCON.2020.024.

Phillip A. Laplante M.K. 2022. What Every Engineer Should Know about Software Engineering, 2nd Edition, CRC Press, Boca Raton.

Rampini L. and Re Cecconi F. (2022). Artificial intelligence in construction asset management: a review of present status, challenges and future opportunities, Journal of Information Technology in Construction, vol. 27, pp. 884-913, DOI: 10.36680/j.itcon.2022.043.

Ramzi E., Audebert N., Thome N., Rambour C. and Bitot X. Hierarchical Average Precision Training for Pertinent Image Retrieval, Springer Nature Switzerland, pp. 250-66.

Russo D. (2024). Navigating the Complexity of Generative AI Adoption in Software Engineering, ACM Trans. Softw. Eng. Methodol., vol. 33, no. 5, p. Article 135, DOI: 10.1145/3652154.

Samuelson O. and Stehn L. (2023). Digital transformation in construction – a review, Journal of Information Technology in Construction, vol. 28, pp. 385-404, DOI: 10.36680/j.itcon.2023.020.

Sanic M.T., Guo C., Leng J., Guo M. and Ma W. Towards Reliable AI Applications via Algorithm-Based Fault Tolerance on NVDLA, pp. 736-40, DOI: 10.1109/MSN57253.2022.00120.

Sanni-Anibire M.O., Zin R.M. and Olatunji S.O. (2021). Machine learning - Based framework for construction delay mitigation, Journal of Information Technology in Construction, vol. 26, pp. 303-18, DOI: 10.36680/j.itcon.2021.017.

Santhanam P., Farchi E. and Pankratius V. (2019). Engineering Reliable Deep Learning Systems, ArXiv, vol. abs/1910.12582.

Saoudi O., Singh I. and Mahyar H. (2023). Autonomous Vehicles: Open-Source Technologies, Considerations, and Development, Advances in Artificial Intelligence and Machine Learning, vol. 3, no. 1, pp. 669-92, DOI: 10.54364/AAIML.2023.1145.

Sari M., Ali Berawi M., Zagloel T.Y., Madyaningarum N., Miraj P., Pranoto A.R. et al. (2023). MACHINE LEARNING-BASED ENERGY USE PREDICTION FOR THE SMART BUILDING ENERGY MANAGEMENT SYSTEM, Journal of Information Technology in Construction, vol. 28, pp. 622-45, DOI: 10.36680/j.itcon.2023.033.

Saridaki M. and Haugbølle K. (2022). Recognising diversity of data management approaches towards lifecycle costing through personas, Journal of Information Technology in Construction, vol. 27, pp. 1042-59, DOI: 10.36680/j.itcon.2022.051.

Singh J. and Anumba C.J. (2022). Real-time pipe system installation schedule generation and optimization using artificial intelligence and heuristic techniques, Journal of Information Technology in Construction, vol. 27, pp. 173-90, DOI: 10.36680/j.itcon.2022.009.

Sutrisna M. 2009. Research Methodology in Doctoral Research: Understanding the Meaning of Conducting Qualitative Research.

Tallgren M.V., Roupé M., Johansson M. and Bosch-Sijtsema P. (2020). Bim-tool development enhancing collaborative scheduling for pre-construction, Journal of Information Technology in Construction, vol. 25, pp. 374-97, DOI: 10.36680/j.itcon.2020.022.

Tamburri D.A. Sustainable MLOps: Trends and Challenges, pp. 17-23, DOI: 10.1109/SYNASC51798.2020.00015.

Tian C., Chen Y., Zhang J. and Feng Y. (2024). Integrating domain knowledge with deep learning model for automated worker activity classification in mobile work zones, Journal of Information Technology in Construction, vol. 29, pp. 264-80, DOI: 10.36680/j.itcon.2024.013.

Treude C. Navigating Complexity in Software Engineering: A Prototype for Comparing GPT-n Solutions, pp. 1-5, DOI: 10.1109/BotSE59190.2023.00008.

Tsai T.H. and Wu P.H. (2024). Design and implementation of deep learning-based object detection and tracking system, Integration, vol. 99, p. 102240, DOI: 10.1016/j.vlsi.2024.102240.

Tsui F., Karam O. and Bernal B. 2022. Essentials of Software Engineering, Jones & Bartlett Learning.

Uggla G. and Horemuz M. (2020). Identifying roadside objects in mobile laser scanning data using image-based point cloud segmentation, Journal of Information Technology in Construction, vol. 25, pp. 545-60, DOI: 10.36680/J.ITCON.2020.031.

Vermesan O., Nava M.D. and Debaillie B. 2022. Embedded Artificial Intelligence: Devices, Embedded Systems, and Industrial Applications.

Vogelsang A. and Borg M. Requirements Engineering for Machine Learning: Perspectives from Data Scientists, pp. 245-51, DOI: 10.1109/REW.2019.00050.

Wang C., Iyengar S.S. and Sun K. 2023. AI embedded assurance for cyber systems.

Wang J., Antwi-Afari M.F., Tezel A., Antwi-Afari P. and Kasim T. (2024). Artificial intelligence in cloud computing technology in the construction industry: a bibliometric and systematic review, Journal of Information Technology in Construction, vol. 29, pp. 480-502, DOI: 10.36680/j.itcon.2024.022.

Yang D., Ngoc K.M., Shin I. and Hwang M. (2023). DPReLU: Dynamic Parametric Rectified Linear Unit and Its Proper Weight Initialization Method, International Journal of Computational Intelligence Systems, vol. 16, no. 1, DOI: 10.1007/s44196-023-00186-w.

Yu Y., Rashidi M., Samali B., Mohammadi M., Nguyen T.N. and Zhou X. (2022). Crack detection of concrete structures using deep convolutional neural networks optimized by enhanced chicken swarm algorithm, Structural Health Monitoring, vol. 21, no. 5, pp. 2244-63, DOI: 10.1177/14759217211053546.

Yu Y., Rashidi M., Samali B., Yousefi A.M. and Wang W. (2021). Multi-Image-Feature-Based Hierarchical Concrete Crack Identification Framework Using Optimized SVM Multi-Classifiers and D–S Fusion Algorithm for Bridge Structures, Remote Sensing, vol. 13, no. 2, p. 240.

Zeb A., Din F., Fayaz M., Mehmood G. and Zamli K.Z. (2023). A Systematic Literature Review on Robust Swarm Intelligence Algorithms in Search-Based Software Engineering, Complexity, vol. 2023, no. 1, p. 4577581, DOI: 10.1155/2023/4577581.

Zhang Y., Robinson D.K.R., Porter A.L., Zhu D., Zhang G. and Lu J. (2016). Technology roadmapping for competitive technical intelligence, Technological Forecasting and Social Change, vol. 110, pp. 175-86, DOI: 10.1016/j.techfore.2015.11.029.

Zhao J. 2020. Quantum Software Engineering: Landscapes and Horizons.

Zhou X. and Flood I. (2024). Optimization and evaluation of a neural network based policy for real-time control of construction factory processes, Journal of Information Technology in Construction, vol. 24, pp. 84-98, DOI: 10.36680/j.itcon.2024.005.

Zhou Y., Yu Y. and Ding B. Towards MLOps: A Case Study of ML Pipeline Platform, pp. 494-500, DOI: 10.1109/ICAICE51518.2020.00102.

Zhu H., Liu D., Bayley I., Harrison R. and Cuzzolin F. Datamorphic Testing: A Method for Testing Intelligent Applications, pp. 149-56, DOI: 10.1109/AITest.2019.00018.