**DOI: 10.32347/2412-9933.2026.65.97-106**
UDC 005.32: 005.4

**Artem Yurechko**
ORCID: https://orcid.org/0009-0000-5850-9713
*Taras Shevchenko National University of Kyiv, Kyiv, Ukraine*
Postgraduate student, Department of Management technologies
**Denys Liashenko**
ORCID: https://orcid.org/0009-0007-5590-9587
*Taras Shevchenko National University of Kyiv, Kyiv, Ukraine*
Postgraduate student, Department of Management technologies
**Oleksandr Timinskyi**
ORCID: https://orcid.org/0000-0001-8265-6932
*Taras Shevchenko National University of Kyiv, Kyiv, Ukraine*
PhD, Associated Professor, Department of Management technologies

***Article history:***
*Received: 30.01.2026*
*Accepted: 11.02.2026*
*Published: 26.03.2026*

# ANALYSIS OF IT PROJECT LIFE CYCLE MODELS IN THE FIELDS OF E-COMMERCE AND CONTENT-ORIENTED SYSTEM DEVELOPMENT

***Abstract.*** *The article demonstrates that universal approaches to modelling the IT project life cycle do not sufficiently reflect the domain-specific determinants that shape managerial decision-making in complex digital environments. In conditions of increasing structural and technological complexity, traditional phase-based models require conceptual adaptation to the dominant sources of uncertainty inherent in different classes of IT systems. The study provides a theoretical substantiation of a domain-oriented approach to life cycle modelling and develops specialized models for IT projects in the fields of e-commerce and content-oriented system development. The research analyses contemporary frameworks of IT project life cycle management and identifies their limitations in addressing structural heterogeneity, continuous integration environments, and multidimensional operational constraints. Particular attention is given to the concept of a structural determinant as a system-forming factor that defines phase transition criteria and control mechanisms. On this foundation, a risk-oriented life cycle model is proposed for online e-commerce platforms, where decision-making at phase gates is based on an integral risk indicator that aggregates probability, impact, and scenario parameters. For content-oriented systems, an architecture-oriented model is developed, in which phase transitions depend on an integral assessment of architectural coherence, API consistency, and structural stability of data models. The comparative analysis demonstrates that structural symmetry of the life cycle can be preserved while varying its substantive content in accordance with the functional and architectural characteristics of the system under development. The proposed models enhance the formalization of phase transition mechanisms and improve the justification of managerial decisions across different classes of IT projects. The results establish a methodological foundation for further advancement of domain-oriented life cycle modelling in adaptive digital environments.*

***Keywords: IT project life cycle; domain-oriented approach; e-commerce; content-oriented systems; risk management; architectural coherence; phase gates; project management***

## Introduction

The life cycle of an IT project is traditionally defined as a structured sequence of phases that reflects the logic of creating, implementing, and maintaining a software product. It serves as a conceptual and organizational framework that determines how project activities are coordinated over time, how resources are allocated, and how performance is evaluated. Over recent decades, approaches to life cycle design have undergone substantial transformation. Rigid sequential waterfall models have gradually been replaced by iterative, agile, and hybrid methodologies capable of responding rapidly to changing requirements, technological shifts, and evolving stakeholder expectations.

Despite this methodological evolution, most widely adopted life cycle models remain general in nature. They describe phase sequences and management artefacts in

 

universal terms, assuming that similar structural logic can be applied across different types of IT systems. However, contemporary IT projects increasingly operate in environments characterized by high complexity, continuous deployment, distributed architectures, and persistent integration with external services. In such contexts, uncertainty is not merely an external influence but an inherent property of the system under development.

Digital platforms functioning in continuous operational environments integrate with payment gateways, third-party Application Programming Interfaces (APIs), data analytics services, cloud infrastructures, and security mechanisms. They process large volumes of transactional and behavioral data in real time. These conditions generate multiple layers of uncertainty related to performance stability, data consistency, cybersecurity threats, regulatory compliance, architectural scalability, and service availability. Under such circumstances, the life cycle model performs not only an organizational function but also defines decision-making mechanisms, phase transition criteria, control parameters, and principles of long-term system evolution.

Online e-commerce platforms and content-oriented systems provide illustrative examples of this challenge. Although both categories belong to the broader domain of digital platforms and may share technological components such as web interfaces, APIs, databases, and cloud deployment environments, they differ substantially in the nature of their core processes and structural priorities. E-commerce systems are transaction-driven and critically dependent on reliability, financial integrity, risk control, and uninterrupted service delivery. Content-oriented systems, particularly those based on Headless or API-first architectures, prioritize data model consistency, integration flexibility, omnichannel distribution, and architectural coherence over time.

These distinctions indicate that universal life cycle models do not fully capture the domain-specific constraints and dominant sources of uncertainty inherent in each class of systems. Application of a general framework without adaptation may result in inadequate phase transition criteria, misaligned control mechanisms, or insufficient attention to critical structural determinants. There is therefore a need to reconsider the conceptual foundations of life cycle modelling in order to integrate domain-oriented determinants directly into the structure of phases and decision gates.

Accordingly, development and comparative analysis of life cycle models tailored to specific classes of IT systems becomes necessary. Such an approach replaces formal application of general methodologies with substantiated alignment of phase structures, control indicators, and management criteria with the dominant characteristics of the system under development. By grounding life cycle design in domain-specific factors, it becomes possible to enhance the validity of managerial decisions, strengthen system resilience, and ensure coherent evolution within complex digital environments.

## Problem statement

Contemporary IT projects operate under conditions of significant technical and organizational complexity. Distributed system architectures, numerous integrations, and the requirement for continuous operation create this environment. Under such conditions, the project life cycle model no longer functions as a formal sequence of phases. It becomes a structured instrument for organizing managerial decision-making.

Projects in e-commerce and in the development of content-oriented systems require particular attention. Although both belong to the broader category of complex IT systems, their domain-specific characteristics differ substantially. In e-commerce projects, transactional processes dominate. These projects depend on payment infrastructure integration, continuous service availability, and robust data protection. In contrast, content-oriented systems, including Headless Content Management Systems (Headless CMS), prioritize architectural coherence, stability of Application Programming Interface (API) contracts, manageability of content models, and support for omnichannel content delivery.

Existing IT project life cycle models, including waterfall, spiral, iterative, and agile approaches, generally maintain a universal structure. They rarely account for domain-specific characteristics. In practice, project teams adjust phase structures, transition criteria, and decision-making mechanisms informally according to system type. This situation creates the need to develop domain-oriented life cycle models that reflect the structural features of specific classes of IT projects and formalize the logic of their development.

## Review of recent research and publications

Research on the design and adaptation of IT project life cycle models occupies a central position in both academic and applied studies in project management and software engineering. Classical approaches to life cycle structuring, including the waterfall model described by R. Royce, define a sequential progression of phases from requirements specification to system maintenance. The spiral model proposed by B. Boehm further developed this logic by introducing risk analysis as a core element of each iteration [1]. These approaches established fundamental principles of work decomposition and result control. However, they emerged within relatively stable technological environments.

Standards and project management guidelines have significantly influenced the evolution of life cycle

models. The PMBOK® Guide defines the project life cycle as a set of phases that determine the logic of result creation, while treating risk management as a distinct knowledge area [2]. ISO 31000 defines risk as the effect of uncertainty on objectives and integrates risk management into the overall governance system of an organization [3]. Nevertheless, these documents do not specify how domain-specific characteristics of IT systems affect phase structures and transition mechanisms.

The introduction of agile methodologies has further shaped life cycle organization. The Scrum Guide emphasizes iterative value delivery and team self-organization [4]. DevOps practices and continuous integration aim to shorten development cycles and reduce technical risks through automation [5]. Studies on hybrid models combine sequential and iterative approaches while considering the characteristics of infrastructure or software projects [6]. Even so, most of these works describe the life cycle as a universal framework applicable to diverse system types without substantial structural transformation.

Another research direction focuses on managing IT projects under high uncertainty and risk. Studies on risk-oriented approaches stress the integration of quantitative risk indicators into decision-making processes [7]. However, most publications treat risk management as a functional process within a selected methodology rather than as a structural element that shapes the life cycle itself.

In the context of e-commerce projects, the literature highlights the complexity of payment system integration, cybersecurity requirements, and platform scalability [8]. Research in electronic business indicates that such systems operate under conditions of continuous evolution, which complicates the application of linear life cycle models [9]. Nevertheless, most studies concentrate on business models or technological aspects and do not propose a formalized domain-oriented life cycle structure.

In the field of content-oriented systems and Headless architectures, research primarily addresses architectural principles, the API-first approach, and microservice-based system organization [10]. Publications in software architecture emphasize contract stability, API version management, and data model evolution [11]. Yet these works rarely formalize the influence of architectural characteristics on the structure of the IT project life cycle.

The analysis of scientific sources therefore demonstrates that:

– existing IT project life cycle models maintain a universal structure and rarely adapt to specific system classes;

– risk-oriented approaches function mainly as auxiliary processes rather than as structural foundations of the model;

– architectural characteristics of content-oriented systems remain largely detached from life cycle formation;

– the literature lacks a systematic comparative analysis of domain-oriented life cycle models for different types of IT projects.

These findings justify the development and comparison of life cycle models in which structural determinants consist of risk indicators for e-commerce systems and architectural characteristics for content-oriented systems.

## Purpose of the article

The purpose of this article is to provide a theoretical justification and to develop domain-oriented life cycle models for IT projects in the fields of e-commerce and content-oriented system development. The study also conducts a comparative analysis of these models, taking into account the structural characteristics of the respective system classes.

The research establishes conceptual foundations for constructing life cycle models in which the structural determinants differ by domain. For online e-commerce platforms, risk indicators serve as the primary structural factor. For content-oriented systems, architectural characteristics perform this role. This approach aligns the structure of life cycle phases with the nature of the dominant uncertainty inherent in each project type.

To achieve this purpose, the study addresses the following research objectives:

– to analyze existing approaches to modelling the IT project life cycle and identify their limitations in accounting for domain specificity;

– to determine the key characteristics of e-commerce projects that influence the formation of a risk-oriented life cycle structure;

– to substantiate an architecture-oriented approach to constructing a life cycle model for content-oriented systems, with consideration of API contracts, data models, and omnichannel delivery mechanisms;

– to conduct a comparative analysis of the proposed models in order to identify their common elements and fundamental differences.

## Main research

The development of domain-oriented life cycle models for IT projects requires a shift from a universal description of phase sequences to a framework that accounts for the specific characteristics of the system under development and the conditions of its operation. Traditional approaches to life cycle modelling rely on general principles of work organization and task control. However, these principles do not suffice in the contemporary digital environment. Complex software platforms differ not only in their multi-component

architecture, distributed infrastructure, and integration with external services, but also in the nature of uncertainty that shapes project management logic. The dominant type of uncertainty determines which indicators require priority, which risks demand attention, and which managerial decisions become critical.

In e-commerce projects, transactional reliability, financial accountability, service continuity, personal data protection, and integration with payment and logistics systems assume critical importance. Even minor failures may result in direct financial losses and erosion of user trust. Under such conditions, the life cycle structure must focus not only on achieving functional readiness of the product but also on systematic monitoring and evaluation of risk characteristics at each implementation phase. Decisions regarding transition to the next stage must reflect the acceptability of the current risk level and the system's readiness to operate under high-load conditions.

In content-oriented systems, particularly headless architectures, different factors dominate. These include data model stability, consistency of Application Programming Interface (API) contracts, manageability of integrations, scalability, and the capacity for system evolution without loss of structural integrity. Such systems often operate within omnichannel content delivery environments and interact with multiple client applications. In this context, immediate operational risks play a lesser role than the long-term consequences of architectural decisions. Uncontrolled modification of data structures or incompatibility of interfaces may significantly hinder future system development. Therefore, life cycle logic must incorporate architectural characteristics as a central decision-making factor and establish mechanisms to verify their consistency during design and scaling stages.

This study therefore proposes an approach in which the life cycle model derives from the structural determinant of a specific system class. For online e-commerce platforms, this determinant is the project risk profile, which reflects the manageability and acceptability of uncertainty. For content-oriented systems, architectural coherence and stability of component interaction determine long-term system viability. The following subsections formalize the respective models and compare them in order to identify the shared and distinctive features of the domain-oriented approach.

## Methodological foundations for constructing domain-oriented life cycle models

The methodological foundation for constructing domain-oriented life cycle models rests on the assumption that the structure of project phases must correspond to the nature of the system's dominant uncertainty. Universal approaches define the life cycle as a sequence of initiation, planning, implementation, and closure stages. However, transition criteria between these stages usually remain general and are formalized primarily through the completion of planned tasks.

The proposed approach identifies a structural determinant that influences system stability and controllability. For each class of IT project, a set of indicators reflects the state of this determinant. These indicators serve as criteria for decision-making regarding transitions between phases.

Methodologically, the model construction process includes:

– analysis of the system's domain specificity in order to identify dominant constraints and the type of uncertainty;

– development of an indicator system that quantitatively or qualitatively characterizes the state of the structural determinant;

– integration of these indicators into the phase transition mechanism, where decisions on continuation or adjustment depend on achieving defined threshold values;

– incorporation of cyclical development, since contemporary IT systems evolve continuously and the life cycle assumes a recurring structure.

Table 1 presents the comparative methodological characteristics of domain-oriented approaches to life cycle model construction.

*Table 1 –* ***Methodological characteristics of domain-oriented IT project life cycle models***

| Criterion | E-commerce Projects | Content-Oriented Systems |
|---|---|---|
| Dominant Uncertainty Factor | Transactional, financial, and security risks | Architectural coherence and stability of data models |
| Structural Determinant | Project risk profile | Architectural integrity of the system |
| Nature of Key Managerial Decisions | Assessment of risk acceptability prior to transition to the next phase | Confirmation of architectural stability prior to scaling or integration |
| Type of Control Indicators | Reliability metrics, security metrics, financial loss indicators, Service Level Agreement (SLA) metrics | API consistency, stability of data schemas, compliance with non-functional requirement |
| Risks of Late Changes | Financial losses, erosion of user trust | Loss of compatibility, architectural degradation, increased complexity of system evolution |
| Evolution Logic | Iterative reduction of the risk profile | Controlled architectural evolution |

As Table 1 demonstrates, the general structural logic of project management remains consistent. However, the content of phase transition criteria varies substantially according to the domain specificity of the system. In e-commerce projects, control and acceptability of risk levels determine transition decisions. In content-oriented systems, architectural coherence and stability of component interaction assume primary importance.

Thus, the domain-oriented life cycle model preserves the overall structural logic of project management while specifying the content of phases and their completion criteria according to system characteristics. This structure enables the development of models that are symmetrical in form yet distinct in substance for different classes of IT projects.

The proposed methodological framework provides a common basis for the subsequent development of a risk-oriented life cycle model for e-commerce systems and an architecture-oriented model for content-oriented systems. This structure allows their comparison without compromising logical consistency.

## Risk-oriented life cycle model for e-commerce IT projects

IT projects that develop online e-commerce platforms operate in environments characterized by intensive transactional activity, complex integration dependencies, and strict reliability and security requirements. These systems interact with payment gateways, logistics operators, Customer Relationship Management (CRM) platforms, analytics services, and external Application Programming Interfaces (APIs), thereby forming a multi-layered network of dependencies. In such an environment, a technical incident or security breach directly results in financial losses, regulatory penalties, erosion of user trust, and reputational damage. Uncertainty in e-commerce projects therefore assumes a complex nature and encompasses technical, financial, operational, and legal dimensions.

In response to these conditions, the proposed life cycle model treats risk not as an auxiliary control mechanism but as a structural determinant of project management. The model combines a process-based organization of development phases with a formalized mechanism for evaluating the project risk profile. The results of this evaluation directly influence decisions regarding transitions between phases. Figure 1 presents the generalized three-level structure of the model, which includes the process level, the risk evaluation level, and the decision-making level.

The life cycle comprises five interrelated phases organized in a cyclical structure. During the initiation phase, the project defines strategic objectives, clarifies the business context, identifies key stakeholders, and performs initial risk identification. Particular attention focuses on market, regulatory, and financial threats that may affect the feasibility of launching the initiative.

The planning and design phase specifies functional and non-functional requirements, develops the architectural concept, and plans integrations and releases. At this stage, the project assesses architectural, integration-related, and resource risks that may influence subsequent implementation.

The development phase includes implementation of functional modules, integration with external services, and testing activities. Technical, operational, and schedule-related risks dominate this phase.

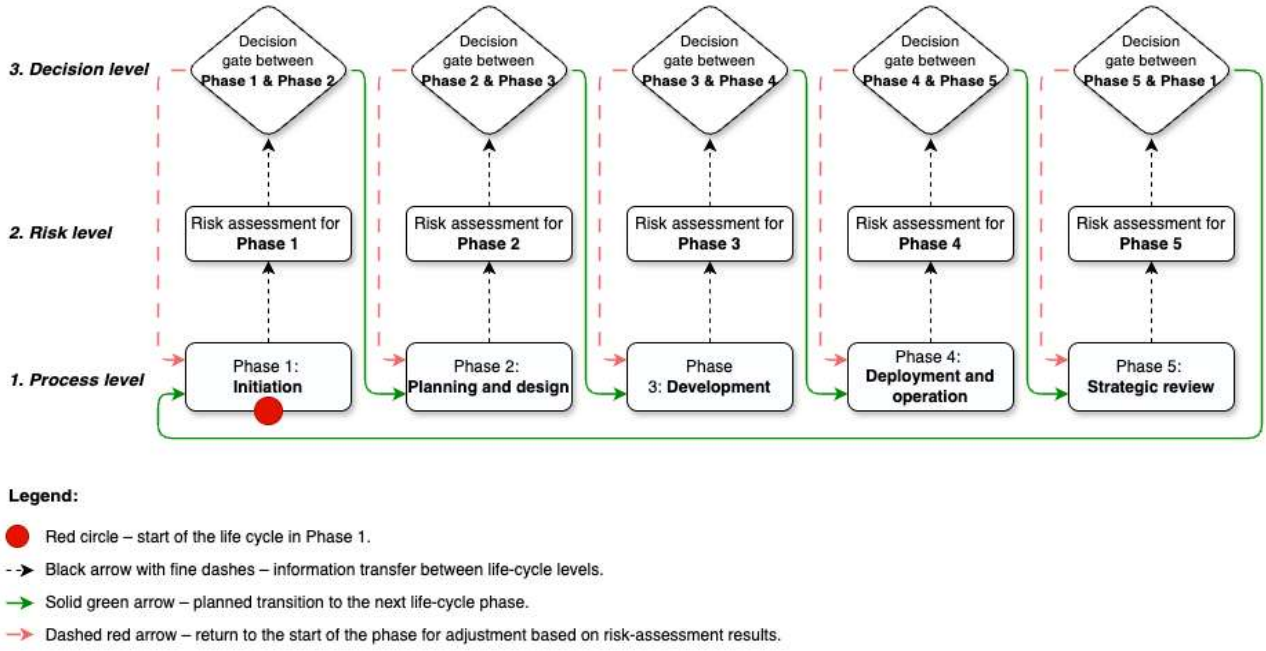


*Figure 1 – Risk-oriented life cycle model of an IT project for the development of an online e-commerce platform*

The deployment and operation phase involves system release into the production environment and monitoring of availability, performance, and security indicators in accordance with established Service Level Agreements (SLAs).

During the final strategic review phase, the project aggregates operational results, analyses the realized risk profile, and determines the feasibility of further development, modernization, or project termination. This decision may initiate a new life cycle iteration. Given the cyclical nature of e-commerce platform development, the risk evaluation mechanism must operate as a formalized condition for phase transitions and must directly influence managerial decisions.

The formalization of the risk management mechanism introduces an integral indicator of risk impact for each life cycle phase. Let phase $k$ contain a set of identified risk $R_k$. For each risk $i \in R_k$ the model defines:

– $p_{ik}$ – the estimated probability of risk occurrence;

– $L_{ik}$ the estimated magnitude of potential losses in the event of realization;

– $w_{ik}$ – the weight coefficient reflecting the relative importance of the risk within the given phase.

The integral risk indicator for phase $k$ is defined as:

$$RE_k = \sum_{i \in R_k} w_{ik}\, p_{ik}\, L_{ik}. \quad (1)$$

For each phase, the model establishes an admissible threshold value $RE_k^*$ for the integral risk indicator. Additional constraints may apply to specific critical risks. Transition to the subsequent phase is permitted if the following condition holds:

$$RE_k \leq RE_k^*. \quad (2)$$

If the defined boundary values for parameters $p_{ik}$ or $L_{ik}$ remain within acceptable limits for risks of elevated significance. If the threshold values are exceeded, corrective actions must reduce the risk profile or refine project implementation parameters.

The project state at phase $k$ is represented by the vector:

$$x_k = (C_k, T_k, Q_k, S_k, RE_k), \quad (3)$$

where $C_k$ denotes actual or forecast costs, $T_k$ denotes task duration, $T_k$ represents an aggregated quality indicator, $S_k$ denotes a vector of operational metrics (relevant for later phases), and $RE_k$ represents the integral risk indicator.

The composition and evaluation methods of these components may vary by phase. Early-stage assessments rely primarily on expert or forecast estimations. Later stages rely on measured values and statistical data.

The decision at the phase gate is formalized as a discrete function:

$$\delta_k = f(x_k), \quad (4)$$

where $\delta_k$ belongs to the set of admissible managerial decisions.

The set of admissible managerial decisions includes transition to the next phase, conditional transition with restrictions, return for rework, and suspension of activities.

Since risk assessment occurs under conditions of uncertainty, the model applies a scenario-based approach. For each scenario – optimistic, baseline, and pessimistic – the model assigns scenario-specific parameter values $p_{ik}^{(s)}$, $L_{ik}^{(s)}$, and, where appropriate, $w_{ik}^{(s)}$. The integral risk indicator for scenario $s$ is defined as:

$$RE_k^{(s)} = \sum_{i \in R_k} w_{ik}^{(s)}\, p_{ik}^{(s)}\, L_{ik}^{(s)}. \quad (5)$$

Analysis of the dynamics of $RE_k^{(s)}$ across scenarios enables refinement of threshold values $RE_k^*$ and adaptation of phase transition rules to the specific characteristics of a given online platform. In this manner, risk integrates directly into the life cycle management mechanism and functions as a formalized condition for decision-making regarding subsequent project evolution.

Thus, risk integrates directly into the mechanism of phase transitions and functions as a formal condition for managerial decision-making.

In the proposed model, phase gates represent institutionalized decision points. At these points, the project evaluates whether the current risk profile complies with established admissible thresholds and determines the feasibility of progressing to the subsequent phase or adjusting project parameters. Table 2 presents a generalized representation of the decision-making logic applied at phase gates. These decision points establish whether to proceed to the next phase, return for refinement of previous decisions, adjust the architecture, or modify the release scope.

Compared with the waterfall model, in which risk control does not integrate into phase transition mechanisms, and the spiral model, in which risk remains a conceptual element rather than a formalized system of indicators and thresholds, the proposed model provides quantitative justification for managerial decisions. Agile approaches, including Scrum, Kanban, and DevOps, reduce technical risks through iteration and automation. However, they do not introduce an aggregated risk criterion as a formal condition for phase transition. In the proposed model, risk becomes a systemic management parameter that determines the evolutionary logic of the e-commerce platform life cycle.

*Table 2 –* ***Phase gates of the risk-oriented life cycle model***

| Phase Gate | Primary Managerial Question | Key Roles | Typical Decision Options |
|---|---|---|---|
| Phase 1 – Phase 2 | Is the initial risk profile acceptable? | Client, Product Owner, Project Manager | Initiate planning; refine objectives; terminate the initiative |
| Phase 2 – Phase 3 | Are the architecture and release plan viable under the current risk profile? | Architect, Chief Technology Officer (CTO), Security Specialist | Begin development; adjust the architecture; limit release scope |
| Phase 3 – Phase 4 | Is the version ready for deployment considering technical and security risks? | Technical Lead, DevOps Engineer, Quality Assurance (QA) | Proceed with release; conduct additional testing; postpone release |
| Phase 4 – Phase 5 | Has the project achieved an acceptable level of stability and risk? | Project Manager, Site Reliability Engineer (SRE), Business Representatives | Continue operation; strengthen monitoring; initiate review |
| Phase 5 – Phase 1 | Which development scenario is appropriate given accumulated risks? | Senior Management, Product Owner | Launch a new development cycle; modernize; maintain or terminate the project |

## Architecture-oriented life cycle model for content-oriented IT systems

Content-oriented systems, including Headless solutions, operate in environments characterized by omnichannel data delivery, API-first interaction, and continuous evolution of content structures. Unlike transactional e-commerce platforms, where control of operational and financial risks constitutes the primary concern, content-oriented systems depend fundamentally on architectural coherence and the system's capacity for scalable evolution without loss of integrity.

Such systems exhibit a high degree of integration dependency. Modifications to data structures, API contracts, or authentication mechanisms may trigger cascading disruptions across client applications and external services. In this context, life cycle management must prioritize the stability of architectural decisions, the consistency of data models, and compliance with non-functional requirements.

The proposed model preserves a five-phase life cycle structure – initiation, architectural design, development, deployment and stabilization, and strategic architectural review. However, the mechanism governing transitions between phases relies on an integral evaluation of architectural coherence rather than on risk thresholds.

To formalize this approach, the model introduces an integral indicator of architectural coherence at phase $k$, denoted as $A_k$. For each phase, the model defines a set of controlled architectural characteristics $M_k$. These characteristics may include data schema consistency, stability of Application Programming Interface (API) contracts, test coverage level, compliance with Service Level Objectives (SLOs), and technical debt control.

For each characteristic $j \in M_k$, the model assigns a normalised indicator value $a_{jk} \in [0,1]$ and a weight coefficient $v_{jk}$, which reflects its relative importance. The integral architectural coherence indicator is defined as:

$$A_k = \sum_{j \in M_k} v_{jk} a_{jk}. \quad (6)$$

For each phase, the model establishes a minimum admissible value VV, which represents the required level of architectural stability prior to transition to the next stage. The phase transition condition is formulated as:

$$A_k \geq A_k^* \quad (7)$$

If the threshold value is not achieved, corrective actions are initiated. These actions aim to refine the architecture, stabilize API contracts, or eliminate structural inconsistencies.

The system state at phase $k$ may be represented by the vector:

$$y_k = (C_k, T_k, Q_k, A_k), \quad (8)$$

where $C_k$ denotes costs, $T_k$ denotes duration, $Q_k$ denotes an implementation quality indicator, and $A_k$ denotes the integral architectural coherence indicator.

The decision at architectural phase gates is determined as a function of the current system state and is made with reference to the established threshold level $A_k^*$.

Thus, architectural coherence integrates into the life cycle management mechanism and functions as a formalized criterion for phase transitions.

A generalized description of architectural phase gates is presented in Table 3.

*Table 3 –* ***Architectural phase gates of the life cycle of content-oriented systems***

| Phase gate | Key architectural question | Key roles | Typical managerial decisions |
|---|---|---|---|
| Initiation – Architectural design | Does the target data model and the API-first interaction principle correspond to project objectives? | Architect, Product Owner | Approve architectural concept; refine the model |
| Architectural design – Development | Are API contracts and data schemas stable? | Architect, Technical Lead | Proceed with implementation; conduct contract revision |
| Development – Deployment | Is architectural coherence ensured at the implementation and test coverage levels? | Technical Lead, Quality Assurance (QA), DevOps Engineer | Authorize release; intensify testing |
| Deployment – Stabilization | Do delivery indicators comply with established Service Level Objectives (SLOs)? | DevOps Engineer, Site Reliability Engineer (SRE) | Continue scaling; conduct optimization |
| Stabilization – Strategic review | Does the system ensure long-term architectural evolution? | Chief Technology Officer (CTO), Architect | Initiate a new development cycle; conduct refactoring |

The distinction between this model and the risk-oriented model lies in the change of structural determinant. In e-commerce projects, the central concern is control of acceptable risk levels. In content-oriented systems, the decisive condition is attainment of a sufficient level of architectural stability and coherence. This difference ensures methodological symmetry in structural form while preserving substantive divergence in managerial logic.

## Comparative analysis of the models

The developed life cycle models share a common structural foundation yet differ in their dominant managerial determinant and in the decision-making mechanism applied at phase gates. Both models adopt a five-phase structure and incorporate a formalized criterion for transition between stages. However, their substantive logic derives from the nature of the system under development and from the dominant source of uncertainty that shapes managerial priorities.

In the risk-oriented model for e-commerce systems, the structural determinant is the integral risk indicator. Transition between phases is permitted only when the level of risk exposure remains within acceptable limits. In this configuration, uncertainty is interpreted primarily as measurable exposure to financial, operational, and security threats. The management process therefore focuses on quantifying potential impact, constraining unacceptable deviations, and maintaining stability of the transactional environment. Phase gates act as control filters that prevent escalation of cumulative risk and ensure continuity of business operations.

In contrast, the architecture-oriented model for content-oriented systems relies on the integral evaluation of architectural coherence as the key transition criterion. Here, uncertainty manifests not primarily as immediate operational risk but as the possibility of structural degradation, inconsistency of data models, or instability of API contracts. Phase transitions depend on achieving a sufficient level of architectural stability before system expansion, scaling, or integration. Managerial logic thus prioritizes structural consistency, maintainability, and long-term evolutionary resilience over short-term risk containment.

The distinction between the models becomes particularly evident when examining the functional role of phase gates. In the risk-oriented configuration, phase gates operate as quantitative checkpoints that assess admissibility of risk exposure. In the architecture-oriented configuration, they function as structural validation mechanisms that confirm coherence of system design and readiness for further evolution. Although both approaches preserve a symmetrical five-phase structure, the evaluative content embedded within each phase transition reflects fundamentally different managerial concerns.

This comparison highlights that structural symmetry of the life cycle does not imply uniformity of decision logic. Instead, the dominant domain determinant shapes the interpretation of performance indicators, corrective actions, responsible roles, and the temporal horizon of evaluation. Table 4 summarizes the comparative characteristics of the two models and illustrates the divergence in managerial focus despite their shared formal structure.

The comparison demonstrates that, despite the shared five-phase structure, the content of managerial decisions and the criteria governing phase transitions depend substantially on domain specificity. In both approaches, phase gates operate as formalized mechanisms for monitoring project status. However, the set of evaluated parameters and the logic of their interpretation differ.

For online e-commerce platforms, managerial focus centers on acceptable risk exposure and prevention of adverse financial and operational consequences. For content-oriented systems, the decisive objective is preservation of architectural integrity and stability of integration interactions within a dynamic environment.

*Table 4 –* ***Comparative characteristics of domain-oriented life cycle models***

| Criterion | Risk-oriented model | Architecture-oriented model |
|---|---|---|
| Structural determinant | Integral risk indicator | Integral architectural coherence indicator |
| Dominant uncertainty | Transactional, financial, operational | Structural, integration-related, evolutionary |
| Primary transition criterion | $RE_k \leq RE_k^*$ | $RE_k \leq RE_k^*$ |
| Management focus | Minimization of potential losses | Assurance of stability and scalability |
| Key roles at phase gates | Client, Project Manager, Security Specialist | Architect, Technical Lead, DevOps Engineer |
| Typical corrective actions | Refactoring, API stabilization, architectural optimization | Refactoring, API stabilization, architectural optimization |
| Evaluation horizon | Short- and medium-term | Medium- and long-term |

The domain-oriented approach therefore adapts the general life cycle logic to different classes of IT projects without altering its structural form. At the same time, it refines the substantive content of decision criteria applied at phase transitions.

## Scientific novelty of the obtained results

The study has produced the following scientific contributions:

– The approach to modelling the IT project life cycle has been refined through the introduction of a domain-oriented principle for defining phase transition criteria. This principle relies on identification of a structural determinant specific to the relevant domain.

– The concept of formalizing managerial decision-making at phase gates has been further developed through integration of integral system state indicators as mandatory conditions for transition between life cycle phases.

– A system of interrelated life cycle models has been proposed for different classes of IT projects, namely e-commerce platforms and content-oriented systems. This system preserves structural symmetry while allowing variability in substantive content according to the nature of dominant uncertainty.

– The study substantiates the use of integral indicators – specifically the risk profile and architectural coherence – as formalized criteria for phase transitions, thereby enhancing controllability and predictability of IT project development.

## Conclusions and prospects for further research

The article provides a theoretical justification for a domain-oriented approach to constructing life cycle models for IT projects in the fields of e-commerce and content-oriented system development. The findings demonstrate the necessity of aligning life cycle phase structures with the nature of the dominant uncertainty inherent in each system class.

The research develops a risk-oriented life cycle model for online e-commerce platforms, in which phase transitions depend on an integral risk indicator. It also proposes an architecture-oriented model for content-oriented systems, where decision-making relies on an integral assessment of architectural coherence. Comparative analysis confirms the possibility of preserving structural symmetry of the life cycle while varying its substantive content.

The results support the application of formalized phase transition criteria as an instrument for improving the validity of managerial decisions across different classes of IT projects.

Further research should focus on developing methods for empirical validation of the proposed models in real-world projects, refining the composition of integral indicators with regard to sector-specific characteristics, and extending the domain-oriented approach to additional categories of IT systems.

**Conflict of Interest.** The authors confirm that there are no financial, personal, or other interests that could be considered a potential conflict of interest regarding the publication of this article.

**Funding.** The research was conducted without financial support.

**Data Availability.** All data are available in digital or graphical form within the main text of the manuscript.

**Use of Artificial Intelligence.** The authors confirm that no artificial intelligence tools were used in the creation of this work.

## References

1. Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21 (5), 61–72.
2. Boiko, Ye., Diachenko, Y., Shandra, T., & Yakovenko, V. (2024). Formation of project portfolios in IT companies. *Proceedings of the 5th International Workshop IT Project Management (ITPM 2024)*, 3709, 264–277. https://ceur-ws.org/Vol-3709/paper21.pdf
3. Hillson, D. (2002). Extending the risk process to manage opportunities. *International Journal of Project Management*, 20 (3), 235–240.

4. International Organization for Standardization. (2018). *Risk management – Guidelines* (ISO Standard No. 31000:2018). https://www.iso.org/standard/65694.html

5. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps handbook*. IT Revolution Press.

6. Laudon, K. C., & Traver, C. G. (2023). *E-commerce 2023: Business, technology, society*. Pearson.

7. Newman, S. (2021). *Building microservices* (2nd ed.). O'Reilly Media.

8. Project Management Institute. (2021). *A guide to the project management body of knowledge (PMBOK® Guide)* (7th ed.). https://www.pmi.org/pmbok-guide-standards

9. Richards, M., & Ford, N. (2020). *Fundamentals of software architecture*. O'Reilly Media.

10. Schwaber, K., & Sutherland, J. (2020). *The Scrum guide*. https://scrumguides.org

11. Turban, E., King, D., Lee, J., & Liang, T. (2015). *Electronic commerce: A managerial and social networks perspective*. Springer.

---

**Юречко Артем Олександрович**
ORCID: https://orcid.org/0009-0000-5850-9713
*Київський національний університет імені Тараса Шевченка, Київ, Україна*
Аспірант кафедри технологій управління
**Ляшенко Денис Олександрович**
ORCID: https://orcid.org/0009-0007-5590-9587
*Київський національний університет імені Тараса Шевченка, Київ, Україна*
Аспірант кафедри технологій управління
**Тімінський Олександр Георгійович**
ORCID: https://orcid.org/0000-0001-8265-6932
*Київський національний університет імені Тараса Шевченка, Київ, Україна*
Кандидат технічних наук, доцент, доцент кафедри технологій управління

## АНАЛІЗ МОДЕЛЕЙ ЖИТТЄВОГО ЦИКЛУ ІТ-ПРОЄКТІВ У СФЕРАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ТА РОЗРОБКИ КОНТЕНТ-ОРІЄНТОВАНИХ СИСТЕМ

***Анотація.*** *У статті показано, що універсальні підходи до моделювання життєвого циклу ІТ-проєктів недостатньо відображають доменно-специфічні чинники, які визначають логіку управлінських рішень у складних цифрових середовищах. В умовах зростання структурної та технологічної складності традиційні фазові моделі потребують концептуальної адаптації до домінуючих джерел невизначеності, притаманних різним класам ІТ-систем. У дослідженні здійснено теоретичне обґрунтування доменно-орієнтованого підходу до моделювання життєвого циклу та розроблено спеціалізовані моделі для ІТ-проєктів у сферах електронної комерції та створення контент-орієнтованих систем. Проаналізовано сучасні підходи до управління життєвим циклом ІТ-проєктів та визначено їх обмеження щодо врахування структурної неоднорідності, середовищ безперервної інтеграції та багатовимірних операційних обмежень. Особливу увагу приділено концепції структурної детермінанти як системоутворювального фактору, що визначає критерії фазових переходів і механізми контролю. На цій основі запропоновано ризик-орієнтовану модель життєвого циклу для онлайн-платформ електронної комерції, у межах якої прийняття рішень у фазових «воротах» ґрунтується на інтегральному показнику ризику, що агрегує ймовірність, наслідки та сценарні параметри. Для контент-орієнтованих систем розроблено архітектурно-орієнтовану модель, у якій фазові переходи визначаються інтегральною оцінкою архітектурної узгодженості, стабільності API та структурної цілісності моделей даних. Порівняльний аналіз засвідчив можливість збереження структурної симетрії життєвого циклу за варіативності його змістового наповнення відповідно до функціональних і архітектурних характеристик системи, що розробляється. Запропоновані моделі підвищують рівень формалізації механізмів фазових переходів та обґрунтованість управлінських рішень у різних класах ІТ-проєктів. Отримані результати формують методичну основу для подальшого розвитку доменно-орієнтованого моделювання життєвих циклів у адаптивних цифрових середовищах.*

***Ключові слова: життєвий цикл ІТ-проєкту; доменно-орієнтований підхід; електронна комерція; контент-орієнтовані системи; управління ризиками; архітектурна узгодженість; фазові «ворота»; управління проєктами***

---

**Link to publication**

APA Yurechko, A., Liashenko D., & Timinskyi, O. (2026). Analysis of IT project life cycle models in the fields of e-commerce and content-oriented system development. *Management of Development of Complex Systems*, 65, 97–106, dx.doi.org\10.32347/2412-9933.2026.65.97-106.

ДСТУ Юречко А. О., Ляшенко Д. О., Тімінський О. Г. Аналіз моделей життєвого циклу ІТ-проєктів у сферах електронної комерції та розробки контент-орієнтованих систем. *Управління розвитком складних систем.* Київ, 2026. № 65. С. 97 – 106, dx.doi.org\10.32347/2412-9933.2026.65.97-106.