Module 5 Portfolio Milestone: Design Methodology Report

Alexander Ricciardi

Colorado State University Global

CSC501: Management for the Computer Science Professional

Dr. Brian Holbert

December 14, 2025

Module 5 Portfolio Milestone: Design Methodology Report

A project methodology is an approach or framework that outlines the structure and processes used to execute the project work and deliver the final result (Ucertify, n.d.). This report provides an outline of the processes required for the design methodology, more specifically, Omega.py Project Codebase, as well as a Gantt chart of the full project schedule (44 Weeks), including detailed integration phase activities. The Project Codebase utilizes a hybrid methodology that blends predictive and adaptive approaches. The Predictive (sequential) approach is used where requirements are stable, and compliance to constraints is high, such as for software infrastructure, security, and legal compliance. On the other hand, the adaptive (Agile/iterative) approach is used where uncertainty and complexity are high, such as for developing and integrating AI features and the RAG system.

Project Codebase Overview

The goal of the Project Codebase development is to design and implement an intelligent platform (AI Agent + RAG system) that helps Omega.py software engineers and AI coding agents understand complex software architectures. It utilizes a Neo4j knowledge graph and a custom markup language, DE-ML, to store and extract context such as component relationships from software project code and related data.

Processes Required for the Design Methodology

Approach Selection Process

The processes start by selecting a design methodology that is best suited for the project environment, the project purpose, and the project complexity. For Project Codebase, the selection of the hybrid approach was based on the following project variables:

- Degree of innovation of the project - Project Codebase involves developing a markup language (DE-ML), a custom AI agent, and a RAG system. These technical requirements demonstrated a high degree of innovation, but also significant technical uncertainty due to their novel nature. For such a highly innovative project, an adaptive approach, such as Agile/Scrum approach, is required to address evolving changes associated with innovative projects.

- Regulatory and security constraints - Project Codebase must follow strict security protocols regarding code access, and Personally Identifiable Information (PII) needs to be removed from data due to external AI used. Projects operating within strict oversight environments are better suited for a predictive approach to ensure compliance documentation is generated sequentially (Ucertify, n.d.).

- Project complexity - Project Codebase is highly complex software that integrates a backend/API, including a Neo4j graph database with real-time AI power retrieval and an AI agent performing codebase and context analysis. Additionally, it integrates a frontend VS Code plug-in, including a visual interface illustrating the codebase and structure of the software being developed, as well as an interface for Omega.py engineers to interact with Project Codebase's own AI Agent for code and architecture analysis. The complexity of this project is best handled by an iterative approach (Agile/Scrum), breaking down a large, complex problem into more manageable modular components.

For all the reasons listed above, a hybrid methodology was selected for Project Codebase.

Project Life Cycle Definition

The next phase is to set the project life cycle. The project life cycle is structured into logically related phases that include project activities that end with the completion of all the tasks associated with these activities. The Project Codebase life cycle is divided into four process phases integrated within the Work Breakdown Structure (WBS). Please see Table 1 for an overview of the phases and Figures 1 and 2 for a Gantt chart with a detailed breakdown of the phases, as well as the attached Excel spreadsheet.

Table 1
Hybrid Development Approach

Project Phase	WBS Alignment	Development Approach	Delivery	Key Deliverables
Inception/ Planning	Phase 1 Initial Phase 2	Predictive/ Sequential	Single Document	Establish the baselines.
Core AI Delivery	Phase 3 Phase 4	Adaptive/Iterative	Periodic 2-week Sprints	MVP (Minimum Viable Product) increments.
Integration / Deployment	Phase 5	Hybrid/Incremental	Multiple Modules/ Features)	Release tested functionality incrementally (e.g., Auth Module 4.1.1) to users.
Maintenance/Operations	Post-Project	Sustaining Operations	Continuous/ On-Demand	Feature refinement (Product Backlog Grooming) and continuous defect repair as part of the product life cycle.

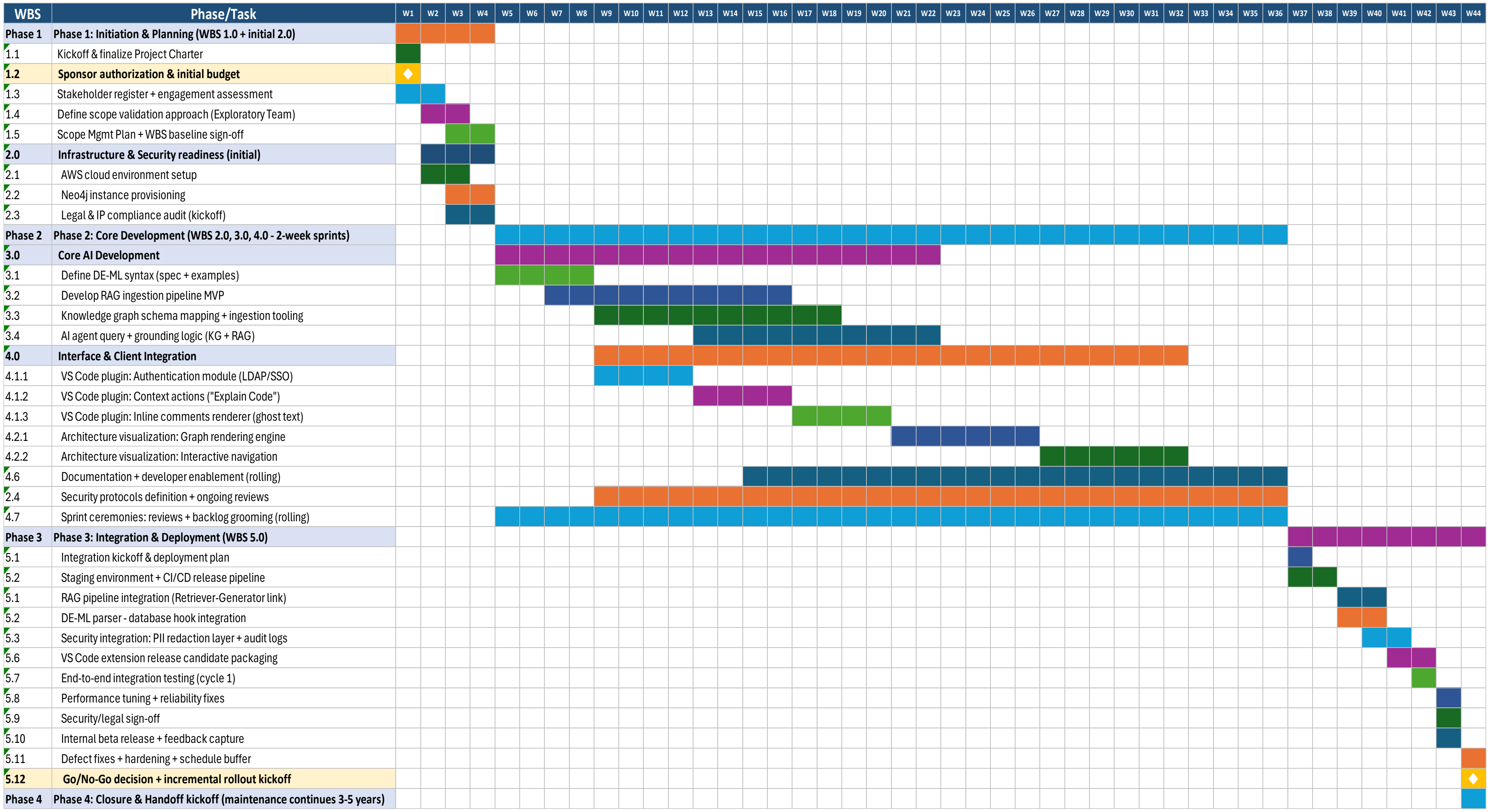
Note: The table illustrates the hybrid development approach integrated within the WBS. From Module 4 Portfolio.

Project Schedule: Gantt Chart (44 Weeks)

A Gantt Chart is a horizontal bar chart used in project management to illustrate a project schedule. It is a visual representation of the project's Work Breakdown Structure (WBS) over time. Figure 1 illustrates the full 44-week project schedule, with a focus on the Integration Phase (WBS 5.0) tasks.

Figure 1

Gantt Chart Full Project Timeline

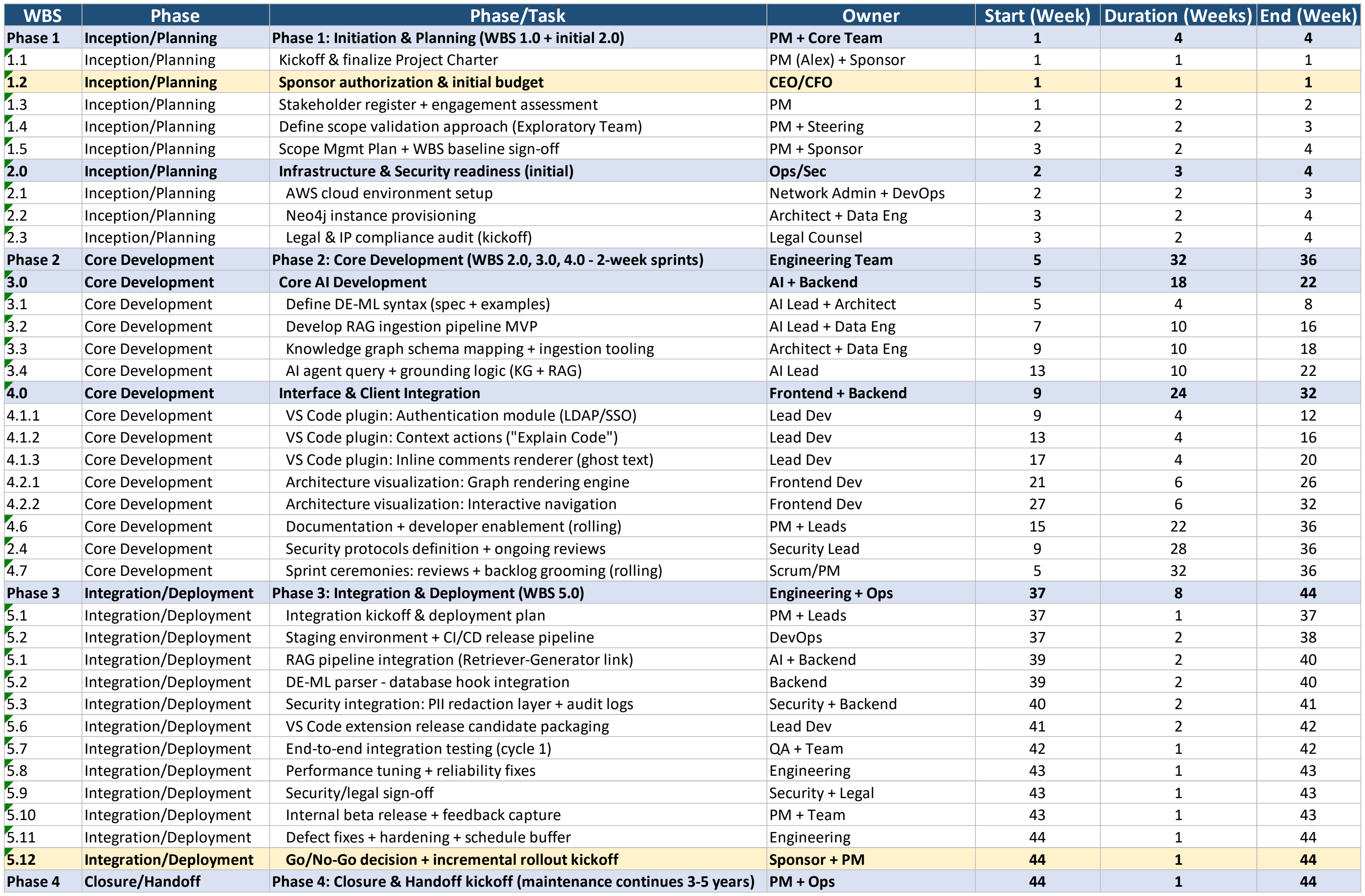


WBS	Phase/Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22	W23	W24	W25	W26	W27	W28	W29	W30	W31	W32	W33	W34	W35	W36	W37	W38	W39	W40	W41	W42	W43	W44
Phase 1	**Phase 1: Initiation & Planning (WBS 1.0 + initial 2.0)**	■	■	■	■																																								
1.1	Kickoff & finalize Project Charter	■																																											
1.2	**Sponsor authorization & initial budget**	◆																																											
1.3	Stakeholder register + engagement assessment	■	■																																										
1.4	Define scope validation approach (Exploratory Team)		■	■																																									
1.5	Scope Mgmt Plan + WBS baseline sign-off			■	■																																								
2.0	**Infrastructure & Security readiness (initial)**		■	■	■																																								
2.1	AWS cloud environment setup		■	■																																									
2.2	Neo4j instance provisioning			■	■																																								
2.3	Legal & IP compliance audit (kickoff)			■	■																																								
Phase 2	**Phase 2: Core Development (WBS 2.0, 3.0, 4.0 - 2-week sprints)**					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■								
3.0	**Core AI Development**					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■																						
3.1	Define DE-ML syntax (spec + examples)					■	■	■	■																																				
3.2	Develop RAG ingestion pipeline MVP							■	■	■	■	■	■	■	■	■	■																												
3.3	Knowledge graph schema mapping + ingestion tooling									■	■	■	■	■	■	■	■	■	■																										
3.4	AI agent query + grounding logic (KG + RAG)													■	■	■	■	■	■	■	■	■	■																						
4.0	**Interface & Client Integration**									■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■												
4.1.1	VS Code plugin: Authentication module (LDAP/SSO)									■	■	■	■																																
4.1.2	VS Code plugin: Context actions ("Explain Code")													■	■	■	■																												
4.1.3	VS Code plugin: Inline comments renderer (ghost text)																	■	■	■	■																								
4.2.1	Architecture visualization: Graph rendering engine																					■	■	■	■	■	■																		
4.2.2	Architecture visualization: Interactive navigation																											■	■	■	■	■	■												
4.6	Documentation + developer enablement (rolling)															■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■								
2.4	Security protocols definition + ongoing reviews									■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■								
4.7	Sprint ceremonies: reviews + backlog grooming (rolling)					■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■								
Phase 3	**Phase 3: Integration & Deployment (WBS 5.0)**																																					■	■	■	■	■	■	■	■
5.1	Integration kickoff & deployment plan																																					■							
5.2	Staging environment + CI/CD release pipeline																																					■	■						
5.1	RAG pipeline integration (Retriever-Generator link)																																							■	■				
5.2	DE-ML parser - database hook integration																																							■	■				
5.3	Security integration: PII redaction layer + audit logs																																								■	■			
5.6	VS Code extension release candidate packaging																																									■	■		
5.7	End-to-end integration testing (cycle 1)																																										■		
5.8	Performance tuning + reliability fixes																																											■	
5.9	Security/legal sign-off																																											■	
5.10	Internal beta release + feedback capture																																											■	
5.11	Defect fixes + hardening + schedule buffer																																												■
5.12	**Go/No-Go decision + incremental rollout kickoff**																																												◆
Phase 4	**Phase 4: Closure & Handoff kickoff (maintenance continues 3-5 years)**																																												■

Note: The figure illustrates Project Codebase's full 44-week timeline using a Gantt chart with a detailed integration phase.

Figure 2

Project Schedule



WBS	Phase	Phase/Task	Owner	Start (Week)	Duration (Weeks)	End (Week)
Phase 1	**Inception/Planning**	**Phase 1: Initiation & Planning (WBS 1.0 + initial 2.0)**	**PM + Core Team**	**1**	**4**	**4**
1.1	Inception/Planning	Kickoff & finalize Project Charter	PM (Alex) + Sponsor	1	1	1
1.2	**Inception/Planning**	**Sponsor authorization & initial budget**	**CEO/CFO**	**1**	**1**	**1**
1.3	Inception/Planning	Stakeholder register + engagement assessment	PM	1	2	2
1.4	Inception/Planning	Define scope validation approach (Exploratory Team)	PM + Steering	2	2	3
1.5	Inception/Planning	Scope Mgmt Plan + WBS baseline sign-off	PM + Sponsor	3	2	4
2.0	**Inception/Planning**	**Infrastructure & Security readiness (initial)**	**Ops/Sec**	**2**	**3**	**4**
2.1	Inception/Planning	AWS cloud environment setup	Network Admin + DevOps	2	2	3
2.2	Inception/Planning	Neo4j instance provisioning	Architect + Data Eng	3	2	4
2.3	Inception/Planning	Legal & IP compliance audit (kickoff)	Legal Counsel	3	2	4
Phase 2	**Core Development**	**Phase 2: Core Development (WBS 2.0, 3.0, 4.0 - 2-week sprints)**	**Engineering Team**	**5**	**32**	**36**
3.0	**Core Development**	**Core AI Development**	**AI + Backend**	**5**	**18**	**22**
3.1	Core Development	Define DE-ML syntax (spec + examples)	AI Lead + Architect	5	4	8
3.2	Core Development	Develop RAG ingestion pipeline MVP	AI Lead + Data Eng	7	10	16
3.3	Core Development	Knowledge graph schema mapping + ingestion tooling	Architect + Data Eng	9	10	18
3.4	Core Development	AI agent query + grounding logic (KG + RAG)	AI Lead	13	10	22
4.0	**Core Development**	**Interface & Client Integration**	**Frontend + Backend**	**9**	**24**	**32**
4.1.1	Core Development	VS Code plugin: Authentication module (LDAP/SSO)	Lead Dev	9	4	12
4.1.2	Core Development	VS Code plugin: Context actions ("Explain Code")	Lead Dev	13	4	16
4.1.3	Core Development	VS Code plugin: Inline comments renderer (ghost text)	Lead Dev	17	4	20
4.2.1	Core Development	Architecture visualization: Graph rendering engine	Frontend Dev	21	6	26
4.2.2	Core Development	Architecture visualization: Interactive navigation	Frontend Dev	27	6	32
4.6	Core Development	Documentation + developer enablement (rolling)	PM + Leads	15	22	36
2.4	Core Development	Security protocols definition + ongoing reviews	Security Lead	9	28	36
4.7	Core Development	Sprint ceremonies: reviews + backlog grooming (rolling)	Scrum/PM	5	32	36
Phase 3	**Integration/Deployment**	**Phase 3: Integration & Deployment (WBS 5.0)**	**Engineering + Ops**	**37**	**8**	**44**
5.1	Integration/Deployment	Integration kickoff & deployment plan	PM + Leads	37	1	37
5.2	Integration/Deployment	Staging environment + CI/CD release pipeline	DevOps	37	2	38
5.1	Integration/Deployment	RAG pipeline integration (Retriever-Generator link)	AI + Backend	39	2	40
5.2	Integration/Deployment	DE-ML parser - database hook integration	Backend	39	2	40
5.3	Integration/Deployment	Security integration: PII redaction layer + audit logs	Security + Backend	40	2	41
5.6	Integration/Deployment	VS Code extension release candidate packaging	Lead Dev	41	2	42
5.7	Integration/Deployment	End-to-end integration testing (cycle 1)	QA + Team	42	1	42
5.8	Integration/Deployment	Performance tuning + reliability fixes	Engineering	43	1	43
5.9	Integration/Deployment	Security/legal sign-off	Security + Legal	43	1	43
5.10	Integration/Deployment	Internal beta release + feedback capture	PM + Team	43	1	43
5.11	Integration/Deployment	Defect fixes + hardening + schedule buffer	Engineering	44	1	44
5.12	**Integration/Deployment**	**Go/No-Go decision + incremental rollout kickoff**	**Sponsor + PM**	**44**	**1**	**44**
Phase 4	**Closure/Handoff**	**Phase 4: Closure & Handoff kickoff (maintenance continues 3-5 years)**	**PM + Ops**	**44**	**1**	**44**

Note: The figure illustrates Project Codebase's full 44-week schedule with a detailed integration phase.

References

Ucertify (n.d). Lesson 3: Development Approach and Life Cycle Performance. Project Manager Professional (PMP) Based on PMBOK7. Ucertify. ISBN: 978-1-64459-415-5