ISSN 2306-1677 (Print) ISSN 2518-7325 (On-line)
Отримано: 27.04.2026; Прийнято: 25.05.2026; Опубліковано: 30.06.2026;;
УДК 004.9:911.5/.9:528.94 http://dx.doi.org/10.31548/zemleustriy2026.02.02

СТАНДАРТИЗАЦІЯ МЕТОДОЛОГІЇ MSF AGILE З ДОПОМОГОЮ ISO/IEC 24744

В. С. ЧАБАНЮК,
кандидат фізико-математичних наук,
старший науковий співробітник,
E-mail: chab3@i.ua
ORCID: 0000-0002-4731-7895
Інститут географії Національної академії наук України
О. П. ДИШЛИК,
виконавчий директор
E-mail: dyshlyk@geomatica.kiev.ua
ORCID: 0000-0002-8066-3925
ТОВ «Геоматичні рішення»

Анотація. *У роботі пропонується стандартизація методології розроблення програмного забезпечення MSF Agile. Вона здійснюється з допомогою метамоделі методологій із стандарта ISO/IEC 24744. Оригінал MSF Agile називається MSF for Agile Software Development і позначається MSF4ASD. MSF4ASD реалізована в MSF версії 4.0 у 2005 р. в інструменті/технології Visual Studio Team System. Тому тут вона називається прикладом конкретної методології розроблення програмного забезпечення. Без реалізації методологія існує, але її краще називати конкретизованою узагальненою методологією MSF з загрозою втрати практичності.*

З огляду на актуальність методологій Agile, MSF Agile може бути практично корисною реалізацією узагальненої методології MSF. Конкретну методологію MSF Agile можливо отримати двома способами: 1) двохкроковою специфікацією або 2) стандартизацією з наступною специфікацією. У цій статті вибрано другий спосіб – спочатку виконується стандартизація MSF Agile з допомогою метамоделі методологій розроблення із стандарта ISO/IEC 24744. Після цього простіше виконувати практично корисну специфікацію, оскільки задача стає типовою. Реалізація буде можливою з використанням різних інформаційних технологій (ІТ), включаючи ІТ Microsoft. Конкретна методологія може вже безпосередньо використовуватися на практиці.

У трьох основних розділах цієї роботи: 1) вводиться засіб стандартизації – описуються потрібні елементи метамоделі методологій розроблення ПЗ із стандарту ISO/IEC 24744; 2) нагадується інформація про MSF4ASD, яка тут спрощена до MSF Agile; 3) MSF Agile представляється з допомогою елементів метамоделі із стандарта ISO/IEC 24744. Цим самим показується, як зробити

MSF Agile стандартизованою методологією. Потім з неї однокроковою редукцією можливо отримати конкретну методологію, яка може конструктивно задовольнити Каркасний підхід і полегшити перехід до Базованої на Патернах Просторової Інженерії.

Ключові слова: *методологія Microsoft Solutions Framework (MSF) Agile, ISO/IEC 24744 стандартизація MSF Agile*

Актуальність

У статті [1] методологія називалася близькою за смислом, але практичнішою порівняно з підходом. Для пояснення практичності тут використовується друга інтерпретація Каркасного підходу - як узагальненої методології. Пояснюється, як здійснити двохкрокову редукцію від узагальненої до специфічної методології MSF. Як приклад розглядається методологія MSF Agile, яка в оригіналі у 2005 р. називалася MSF for Agile Software Development і позначалася MSF4ASD. Результат першого кроку редукції можливо назвати специфікованою узагальненою методологією MSF Agile. Другий крок редукції дозволяє отримати специфічну методологію MSF Agile. А специфічна методологія може вже безпосередньо використовуватися на практиці.

У даній роботі досліджується клас «Методологій Каркасного підходу до дослідження і проектування ПрІС (просторових інформаційних систем)». Наразі паралельно створюється кілька екземплярів цього класу. Раніше в наших роботах вже згадувався один такий екземпляр - методологія розширення ПрІС. Іншим екземпляром є методологія MSF Agile. Важливим є продемонстроване тут застосування до MSF Agile метамоделі методологій розроблення програмного забезпечення зі стандарту ISO/IEC 24744. Завдяки цьому прикладу можемо говорити про властивості усіх екземплярів даного класу методологій. А з ними - і про методологію Базованої на Патернах Просторової Інженерії (БППрІ). Майбутня методологія БППрІ буде частиною Базованої на Моделях Системної Інженерії [4], у якій буде враховуватися Базована на Моделях Програмна Інженерія у викладі [5].

Аналіз останніх досліджень та публікацій

Елементи обох MSF і Agile методологій описувалися у багатьох джерелах. Щоб краще зрозуміти MSF Agile, ми розбиралися з кожною методологією окремо. Методологія MSF 2.0 описана в монографії [6], методологія MSF 4.0 – в монографії [7]. У статті [2] методологія MSF названа мета-методологією або узагальненою. Її специфікація у 2005 році досягнута з використанням поточної версії Visual Studio Team System. Включена до MSF 4.0 MSF for Agile Software Development позначена як MSF4ASD. Без цієї реалізації її можна назвати специфікованою узагальненою методологією. Узагальненою – через приналежність до MSF 4.0. Специфікованою – через те, що сьогодні існує лише їх опис з можливістю кількох реалізацій, а разом з ними – специфікації (як процесу). Реальним прикладом опису технології MSF Agile є [8], де описується версія MSF4ASD

2005 року. Щоб зробити технологію MSF4ASD актуальною сьогодні, потрібна сучасна реалізація.

З методологіями Agile розробленя ПЗ розібратися ще важче, ніж з методологією MSF. Багато авторів дуже вільно поводяться з термінологією і називають Agile методологіями Agile методики і методи. Питання виникають одразу при спробі зрозуміти, що таке «методологія розроблення ПЗ». Для прикладу згадаємо переклад статті «Методології, Методики, Методи, Фреймворки – Що до чого?». Його включено як приклад однієї з точок зору спеціаліста з ІТ у статтю [9]. З нього не витікає ясність розуміння. Адже автор пропонує розуміти перелічені у запитанні терміни залежно від контексту. А контекст може бути кожний раз різним.

З цитованого джерела витікає, що першою проблемою є нестабільність термінології щодо методології Agile, яка у різних авторів різна. Неясність відноситься також до «можливих» складових методології. Натомість, ми називаємо Agile методикою, що обгрунтовується посиланням на класичну монографію [10]. Там "Agile" об'єднувала кілька методів Agile. У 2025 р. предмети цих понять були названі складовими ешелонів сутностей Каркасного підходу до поводження з ПрІС (включаюми їх створення). «Методика» знаходилася «між» процесними сутностями «методологія» і «метод». Відповідно до цієї ієрархії методи можуть «входити» до методики і тільки так – до методології.

Для початкового розуміння явища «Agile» у контексті методологій розроблення ПЗ ми рекомендуємо джерело [11]. Воно не є науковим, однак є актуальним вступом у проблематику Agile, що охопдює більшість практичних питань. Після нього можливо переходити до серйозніших публікацій. Наприклад, до монографій [10] або [12]; остання перекладена на російську мову. У даній статті безпосередньо використані проста пояснювальна стаття [13], Agile-маніфест [16] і архів з [8].

Ми вже маємо досвід застосування «практик» створення складних ієрархічних ПрІС. Тому можемо досить чітко описати як саму практику (відповідає на питання що?), так і процес (відповідає на питання як?). Де факто вони однозначно існують. А якщо існують і практика і процес, то має існувати і методологія. Вважаємо, що станом на початок 2026 р. її потрібно тільки довизначити і описати. Цього достатньо, щоб аргументовано використовувати термін «підхід» і називати його конструктивним (як і стратегію).

На даний момент наш підхід можемо назвати також «Каркасним підходом розширення (Атласних і ГеоІнформаційних Систем)». Він базується на каркасних методах з двох груп, кожного однорідного стосовно (варіанта) предмета: 1) Концептуальних Каркасів X, 2) Каркасів Рішень X[Y], де квадратні дужки [Y] повідомляють, що Y може не існувати. Хоча сам підхід не можна назвати набором однорідних методів, оскільки в першому випадку предметом є X, а в другому – X[Y]. Загалом, каркасними називаються рішення, отримані застосуванням одного з двох або обох каркасів (причому, довільну кількість разів). Поняття X пояснювалось кілька разів раніше. Поняття X[Y] розбивається на два поняття зі змінними значеннями. Y при його наявності показує залежність від значення (ієрархічної) страти, тому КаРі X[Y] може

називатись так: [Загальний (γ) | Понятійний (β) | Аплікаційний (α) | Операційний (ω)] Каркас Рішень страти Y= γ, β, α, ω, Предмета X.

Маємо зауважити, що результати даної статті показують неоднозначність майбутньої методології Каркасного підходу до поводження з ПрІС. Потрібно врахувати, як мінімум, три точки зору: двохкрокову редукцію (специфікацію) мета-методології MSF Microsoft, «нашу» методологію розширення і «пряме» застосування Базованої на Моделях Програмної Інженерії (БМПІ), яка описана у монографії [5].

З початку століття ми застосовували екземпляри обох Каркасів (КоКа і КаРі) у багатьох проектах створення або експлуатації предметів X і/або XY. Фактично одразу після виявлення перших КаРі розпочалися узагальнення, які можливо об'єднати у два напрямки, однорідні за якимось критерієм: 1) предметним, або 2) процесним.

Щоб не задіювати термін «підхід» до MSF, ми використовуємо термін «мета-методологія». Обидва терміни прийнятні до MSF, однак другий термін дозволяє утримувати увагу на методології. Нагадаємо, що у статті [1] згадувалися три точки зору (інтерпретації) на Каркасний підхід. Дана стаття присвячена розвитку 2-ї точки на Каркасний підхід, однак ми свідомо звужуємо контекст до мета-методології MSF.

Метою роботи є стандартизація методології розроблення ПЗ MSF Agile з допомогою стандарту [3]. Вона повинна доповнити введене раніше поняття «узагальнена методологія поводження з ПрІС» і знадобиться при створенні методології Каркасного підходу до дослідження і/або проектування ПрІС з точки зору її використання у створюваній нами Базованій на Патернах Просторовій Інженерії (БППІ).

Матеріали і методи наукового дослідження

У цій роботі використовуються методи Концептуальних Каркасів (КоКа), Каркасів Рішень (КаРі) і Метамодель методологій розроблення ПЗ із стандарту [3]. Методи називаються системними через їх застосовність до дослідження і/або проектування Просторових інформаційних систем (ПрІС). ПрІС у цій статті розуміються дуже широко. Розрізняються класичні і некласичні ПрІС. Відомими прикладами класичних ПрІС є Електронні Атласи такі як Електронна версія Національного Атласу України (ЕлНАУ). Прикладами некласичних ПрІС є системи Просторової інформаційної діяльності (СПрІД) і Атласні ГеоІнформаційні Системи (АГІС), які вже досліджувались в наших роботах.

Метамоделі корисні для визначення понять, правил та відношень, що використовуються для визначення методологій. Хоча методологію можна описати без явної метамоделі, формалізація основних ідей даної методології є цінною при перевірці її узгодженості або при плануванні розширень чи модифікацій. Хороша метамодель повинна враховувати процес, якого слід дотримуватися, робочі продукти, які потрібно створити, та тих, хто відповідає за те, щоб усе це відбулося. У свою чергу, визначення робочих продуктів, які необхідно розробити, передбачає визначення основних структурних блоків моделювання, з яких вони будуються.

Метамоделі часто використовуються методологічними інженерами для побудови або модифікації методологій. У свою чергу, методології використовуються розробниками для створення продуктів або надання послуг у контексті певних зусиль.

Метамодель, методологія та зусилля (в оригіналі «endeavour», ми перекладаємо цей термін також як «втручання») у стандарті [3] відносяться до трьох різних галузей експертизи, які водночас відповідають трьом різним рівням абстракції та трьом різним наборам фундаментальних понять. Варіантом перекладу «endeavour domain» може бути «домен зусиль» або «домен втручання», якщо скористатися монографією [14] і його «метамоделюванням». Оскільки робота, що виконується розробниками на рівні зусиль, обмежена та спрямовується використовуваною методологією, робота, що виконується методологічним інженером на рівні методології, обмежена та спрямовується обраною метамоделлю. Традиційно ці відношення між «шарами моделювання» (у нас – стратами або ешелонами), які тут називаються «доменами», розглядаються як відношення «екземпляр чогось», у яких елементи в одному шарі (страті/ешелоні) або домені є екземплярами деякого елемента в шарі (страті/ешелоні) або домені нижче.

Сказане ми помістили на загальну схему використання γ | β | α | ωКаРі і отримали Рис. 1. Три домени [3; Figure 1] показані зліва на Рис. 1.

Показаний зліва на Рис. 1 [3; Figure 1] називався Три сфери знань, або домени, які виступають у контексті SEMDM. Стрілки означають «представлено» (The three areas of expertise, or domains, which act as a context for SEMDM. Arrows mean "is represented by"). Сам стандарт пояснює Рис. 2. В оригіналі це був [3; Figure 4], чию назву ми не змінювали.

Щоб показати відповідність конструкцій моделі КаРі з класами метамоделі ISO/IEC 24744 використано схему [15]. На ній ми показали відповідність з пакетами Процеси і Продукти КаРі і отримали Рис. 3,

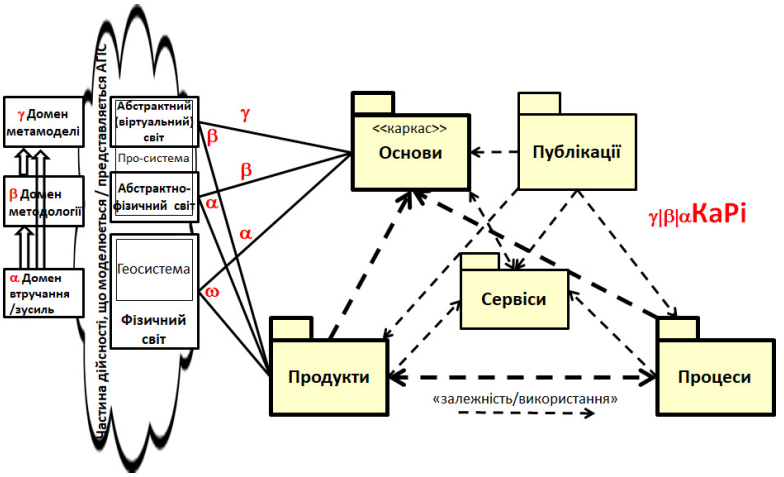


Рис. 1. На схему використання γ | β | α | ωКаРі добавлено [3; Figure 1]

до

а, га
04
02
70
04
59

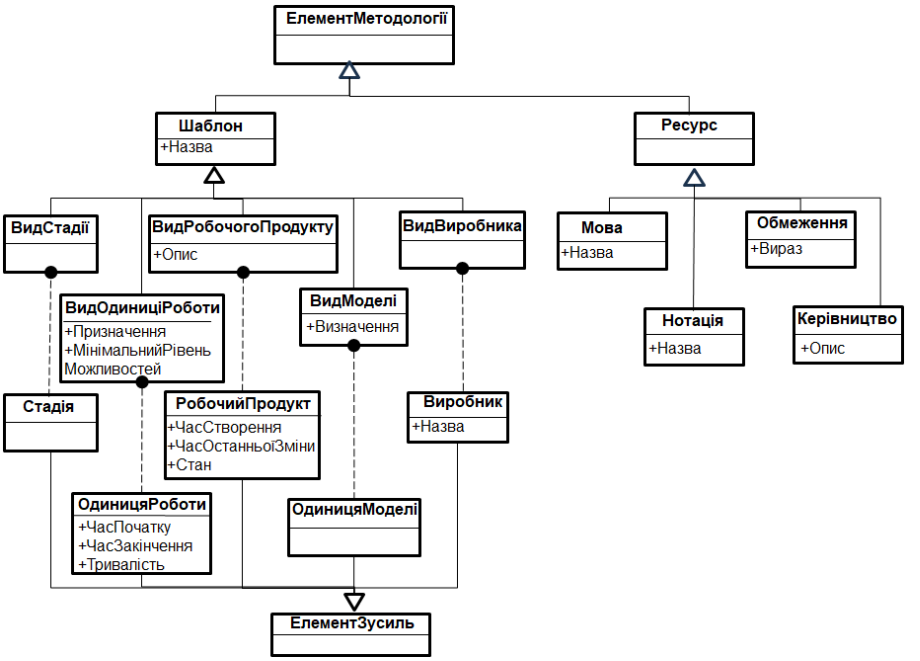


Рис. 2. Абстрактний вигляд SEMDM, що показує основні класи в метамоделі з [3; Figure 4]

не деталізуючи вміст вказаних пакетів. Тут зауважимо, що ми ніколи не забували про користувачів КаРі. Наприклад, опис самого першого КаРі ProSF (Project Solutions Framework) містив призначення пакетів: 1) Пакет Publications призначався для спілкування із зовнішнім середовищем проекта; 2) Пакет Products призначався для розробників продукції проекта; 3) Пакет Processes призначався для менеджменту проекта; 4) Пакет Services призначався для адміністратора, документатора та/або фахівця по навчанню користувачів; 5) Пакет Basics призначався (поелементно) для всіх учасників проекта. Підтримувався архітектором та менеджментом проекта.

Одиниця роботи (Робочий елемент далі) – це завдання, яке потрібно виконати, процес, який потрібно здійснити, або техніка (як процес), яку потрібно застосувати в рамках проекту. Робочий продукт – це проміжний або кінцевий результат, створений під час проекту, що може бути документом, моделлю тощо. Виробник відповідає за виконання одиниць роботи і може бути представлений роллю (набір обов'язків виробника), інструментом (засіб, що дозволяє виробникам виконувати свої обов'язки) та командою (організована група виробників, які мають спільну мету).

Якщо знову згадати відповідність з КаРі, то потрібно зауважити, що у кількох останніх статтях ми розглядаємо Каркасний підхід до поводження з ПрІС, у якому суттєвим є вміст так званих «ешелонів» користувачів ПрІС. До ешелонів входять

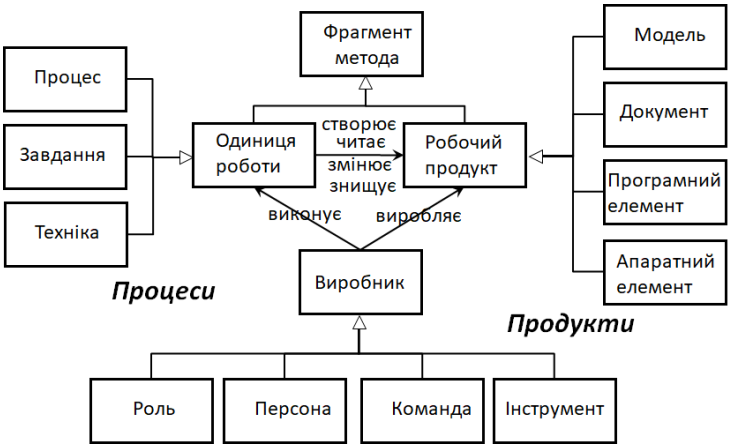


Рис. 3. Часткова концептуальна схема [3] [15] і її відповідність пакетам Процеси і Продукти

так звані «процесні» сутності такі як «методологія», «методика», «метод» тощо. Не вдаючись у подробиці, відмітимо такі основні твердження: 1) ешелони користувачів включають Виробників з Рис. 3; 2) ешелони і їх сутності співвідносяться зі стратами і компонентами тієї чи іншої ПрІС; 3) при конструюванні компонентів страт певної ПрІС використовується відповідний КаРі (γ | β | α | ω).

Об'єднання Процесу і Продукта (5.4 Uniting Process and Product)

При використанні моделей КаРі ніколи не варто забувати про їх важливу характерну ознаку – дуалізм процеси-продукти. У цьому контексті у стандарті [3] є окремий підрозділ 5.4, переклад якого ми наводимо далі.

А саме, більшість існуючих підходів до метамоделювання зосереджені або на процесній, або на моделюючій (тобто продуктовій) стороні методологій. Однак більшість цих підходів пропонують точки з'єднання для «підключення» додаткового, ще невизначеного, компонента повноцінної методології. SEMDM (Software engineering – Metamodel for development methodologies) йде далі, пропонуючи повну метамодель, яка рівномірно охоплює процесні та моделюючі аспекти методологій. Не робити цього було б схоже на спробу визначити дії, які необхідно виконати, не визначивши концепції, на яких ці дії повинні діяти (фокус на процесі), або концепції, які потрібно використовувати, не знаючи, що з ними робити (фокус на моделюванні). Цей підхід має перевагу в тому, що дозволяє глибоко визначити на рівні методології взаємодії між процесом та продуктами, що генеруються ним.

Нарешті, Ресурс (Resource) на Рис. 2 конкретизується (спеціалізується) Мовою (Language) (структура типів одиниць моделі, що зосереджена на певній перспективі моделювання), Нотацією (Notation) (конкретний синтаксис, зазвичай графічний, який може бути використаний для зображення моделей, створених за допомогою певних мов), Керівництвом (Guideline) (указанням, як можна використовувати деякі елементи ме-

тодології) та Обмеженнм (Constraint) (умова, яка виконується або повинна виконуватися в певний момент часу).

Результати дослідження – MSF для гнучкого (Agile) розроблення ПЗ

У розділі враховано монографії [6], [7], архів з [8], стаття [13] і Agile-маніфест [16].

Що таке MSF Agile?

MSF Agile – це методологія розроблення ПЗ, створена Microsoft, щоб допомогти командам керувати програмними проектами з використанням гнучкого підходу. Вона надає каркас робочих потоків (процесів), ролей та ключових дій (діяльності) для проходження командами різних стадій життєвого циклу розроблення ПЗ. Повна назва обговорюваного у цій статті явища – MSF for Agile Software Development (MSF4ASD), яке разом з явищем MSF for CMMI Process Improvement (MSF4CMMI) були «включені» до MSF версії 4.0 у 2005 р.

Для повнішого представлення про MSF4ASD рекомендуємо ознайомитися з матеріалами посилання [8]. Там у файлі ReadMe.txt радиться перейти у каталог Process Guidance (Керівництво Процесом) і відкрити ProcessGuidance.html (Рис. 4). Відомі обмеження: 1) Посилання Project Portal (портал проекту) не працює, коли керівництво процесом використовуються в автономному режимі; 2) Завдання «початку роботи» не відображатимуться в автономній версії; 3) Робочі продукти mpp та xls не заповнюються автоматично, якщо вони не використовуються з Team Foundation Server.

Крім каталога Process Guidance, архів містить каталоги шаблонів (у дужках () наведено їх приклади): Project Management (Development Project Plan.mpp, Project Checklist.xls), Requirements (Scenario Description. doc), Security (Template Sample - Web Application Threat Model.doc), Test (Test Approach.doc). Ми не можемо приділити цим шаблонам більше уваги. У подальшому використаємо тільки елементи Рис. 4.

MSF Agile враховує основні ідеї гнучкого розроблення Agile. Вони формулюються з врахуванням цінностей Agile-маніфесту розроблення ПЗ і його принципів: [16]. Де доціль-

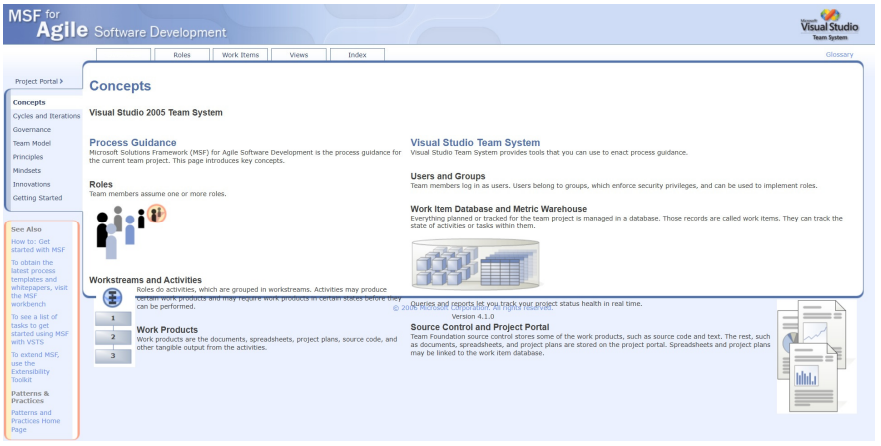


Рис. 4. Сторінка ProcessGuidance.html [8]

но, використаємо також елементи, доступ до яких показано на Рис. 4.

Основні цінності, принципи, мислення і керівництво гнучкого розроблення

Цінності і Принципи Agile-маніфесту

Цінностями знаменитого Agile-маніфесту [16] є Особистості та співпраця, Працюючий продукт, Співпраця із замовником, Готовність до змін. Принципів загалом дванадцять, але вони часто трохи змінюються. Так, в [13] формулюються сім принципів: Співпраця з клієнтами, Сприяти відкритому спілкуванню, Працювати над досягненням спільного бачення, Якість – це щоденна робота кожного, Залишатися гнучким, адаптуватися до змін; Зробити розгортання звичкою, Потік цінностей. В [8] формулюються дев'ять принципів. Доступ до них показаний на Рис. 4 у лівому меню (див. Principles). Щоб передати суть фактично усіх дванадцяти принципів, наведемо переклад першого абзацу і заголовки принципів MSF for Agile Software Development.

MSF for Agile Software Development – це сценарно-керований, контекстно-базований процес гнучкого розроблення ПЗ для створення .NET та інших об'єктно-орієнтованих аплікацій. Він безпосередньо включає практики обробки вимог до якості обслуговування, таких як продуктивність та безпека. Він також контекстно-базований та використовує контекстно-керуємий підхід для визначення способу роботи над проектом. Цей підхід допомагає створити адаптивний процес, який долає граничні умови більшості гнучких процесів розроблення ПЗ, одночасно досягаючи цілей, визначених у баченні проекту.

Заголовки принципів MSF for Agile Software Development: 1) Співпрацюйте з клієнтами (Partner with Customers), 2) Працюйте над досягненням спільного бачення (Work Toward a Shared Vision), 3) Забезпечуйте додаткову цінність (Deliver Incremental Value), 4) Інвестуйте в якість (Invest in Quality), 5) Розширюйте можливості членів команди (Empower Team Members), 6) Встановлюйте чітку відповідальність (Establish Clear Accountability), 7) Вчіться на досвіді (Learn from all experiences), 8) Сприяйте відкритій комунікації (Foster open communications), 9) Залишайтеся гнучкими, адаптуйтеся до змін (Stay agile, adapt to change).

Мислення (Mindsets) і Керівництво (Governance)

Мислення: Якість визначається клієнтом, Гордість за свою роботу, Команда колег, Часте постачання, Готовність навчатися, Раннє уточнення деталей, Якість обслуговування, Громадянська позиція.

Керівництво: Стосується контролю часу та грошей відносно потоку цінності: Чи робимо ми правильно? Чи можемо ми зробити це в рамках часу та бюджету? Чи готові ми до інтеграції? Чи готові ми до розгортання? Чи усвідомлюємо ми цінність?

Крім цінностей, принципів, образу мислення і керівництва MSF Agile визначається тріадою процеси, продукти, виконавці. Далі наведено їх конспективний виклад.

Процеси

Модель процесу MSF добре описана в монографії [6]. Один із варіантів для версії 2.0 наведено на Рис. 5. Фази складалися зі стадій. Моделі наступних версій MSF були варіаціями версії 2.0. Далі наводяться докази для MSF Agile.

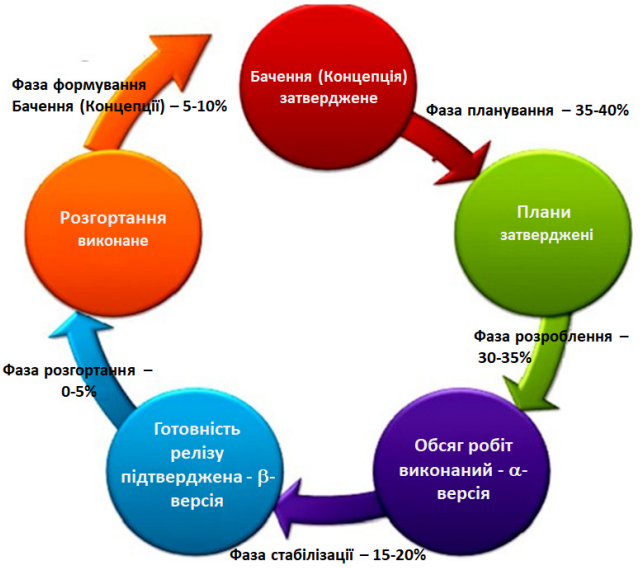


Рис. 5. Модель процесу MSF 2.0

В MSF 4.0 (MSF Agile) модель процесу описується через «робочі потоки». А саме, за [8] ролі виконують дії, які згруповані в робочі потоки. Дії можуть створювати певні робочі продукти та можуть вимагати, щоб робочі продукти були в певних станах, перш ніж їх можна буде виконати. З подальшого опису робочих потоків зрозуміло, що для MSF 4.0 (MSF Agile) справедлива модель процесу MSF 2.0, показана на Рис. 5.

Робочі потоки (Workstreams)

Зафіксувати бачення проекту, Створити вимогу до якості обслуговування, Створити сценарій, Керувати проектом, Планувати ітерацію, Керувати ітерацією, Створити архітектуру рішення, Створити продукт, Виправити помилку, Реалізувати завдання розроблення, Закрити помилку, Тестувати вимогу до якості обслуговування, Тестувати сценарій, Випустити продукт.

Дії по робочих потоках (Activities By Workstream)

Зафіксувати бачення проекту: Написати бачення, Визначити персонажів, Уточнити персонажі.

Створення вимог до якості обслуговування: Мозковий штурм щодо вимог до якості обслуговування, Розробка огляду способу життя (lifestyle), Визначення пріоритетів у списку вимог до якості обслуговування, Написання вимог до якості обслуговування, Визначення цілей безпеки.

Створення сценарію: мозковий штурм сценаріїв, розробка огляду способу життя, визначення пріоритетів списку сценаріїв, написання опису сценарію, створення розкадровки сценарію.

Управління проектом: Огляд цілей, Оцінка прогресу, Оцінка порогових значень тестових показників, Сортування помилок, Виявлення ризиків.

Планування ітерації: Визначення тривалості ітерації, Оцінка сценарію, Оцінка вимог до якості обслуговування, Планування сценарію, Планування вимог до якості обслуговування, Планування розподілу виправлень помилок, Розподіл сценаріїв на завдання, Розподіл вимог до якості обслуговування на завдання.

Управління ітерацією: Моніторинг ітерації, Зменшення ризиків, Проведення ретроспектив.

Створення архітектури рішення: Розбиття системи, Визначення інтерфейсів, Розроблення моделі загроз, Розроблення моделі продуктивності, Створення архітектурного прототипу, Створення архітектури інфраструктури.

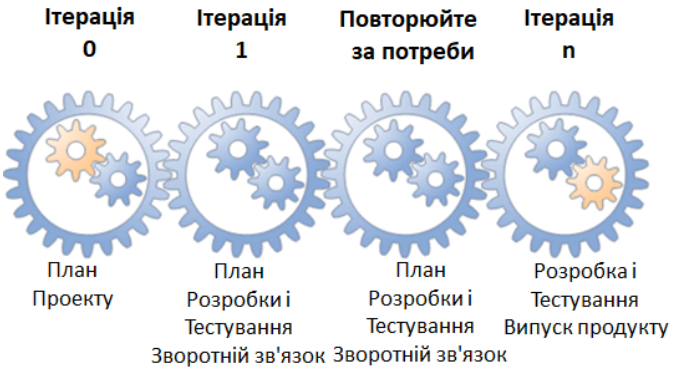
Збірка продукту: Розпочати збірку, Перевірити збірку, Виправити збірку, Прийняти збірку.

Виправлення помилки: Відтворення помилки, Визначення причини помилки, Перепризначення помилки, Вибір стратегії виправлення помилки, Написання коду для виправлення помилки, Створення або оновлення модульного тесту, Виконання модульного тесту, Рефакторинг коду, Перегляд коду.

Реалізація завдання розроблення: Визначення вартості завдання розроблення, Створення або оновлення модульного тесту, Написання коду для завдання розроблення, Виконання аналізу коду, Виконання модульного тесту, Рефакторинг коду, Перевірка коду, Інтеграція змін коду.

Закрити помилку: Перевірити виправлення, Закрити помилку.

Тестування вимоги до якості обслуговування: Визначення підходу до тестування, Написання тестів



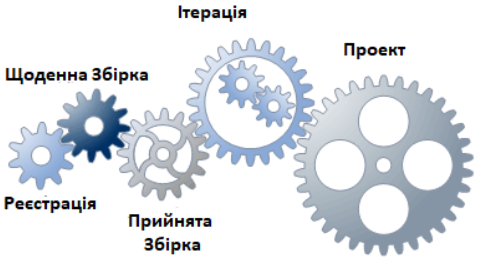


Рис. 6. Цикли та ітерації [8]

продуктивності, Написання тестів безпеки, Написання стрес-тестів, Написання навантажувальних тестів, Вибір та запуск тестового випадку, Відкриття помилки, Проведення дослідного тестування.

Тестування сценарію: Визначення підходу до тестування, Написання валідаційних тестів, Вибір та запуск тестового випадку, Відкриття помилки, Проведення дослідницького тестування.

Випуск продукту: Виконання плану випуску, Перевірка випуску, Створення нотаток до випуску, Розгортання продукту.

Дисципліни, пов'язані з Моделлю процесу MSF (Рис. 5): Вимоги, Планування, Архітектура, Розроблення, Тестування.

Цикли та ітерації / Cycles and Iterations

Плавна інтеграція MSF для Agile у Visual Studio Team System підтримує швидке ітеративне розроблення з безперервним навчанням та вдосконаленням.

- Визначення, розроблення та тестування продукту виконуються ітераціями з перекриттям і це призводить до поступового завершення проекту.
- Різні ітерації мають різний фокус при наближенні проекту до завершення.
- Невеликі ітерації дозволяють зменшити допустиму похибку в оцінках і забезпечити швидкий зворотній зв'язок щодо точності планів проекту.
- Кожна ітерація повинна призвести до стабільної частини всієї системи.

Ітерація 0: Початок проекту, Планування

Ітерація 1: Планування, Розроблення та тестування, Зворотній зв'язок

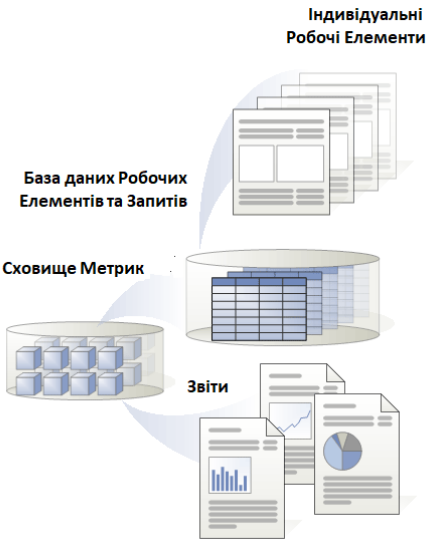


Рис. 7. Елемент Роботи [8]

Повторюйте за потреби: Планування, Розроблення та тестування, Зворотній зв'язок

Ітерація n: Розроблення та тестування, Випуск продукта.

Продукти

Розрізняються Робочі Елементи (Work Items) і Робочі Продукти (Work Products). Перші відносяться до Процесів, другі – до Продуктів. Ми називаємо цей факт «дуалізмом», завдяки чому відмінності між Процесами і Продуктами стають зрозумілішими. Дуалізм справедливий у кожному контексті. Далі розглядаються кілька прикладів.

Робочий Елемент

Робочий Елемент співвідноситься з записом у базі даних, який Visual Studio Team Foundation використовує для відстеження призначення та стану роботи. Процес MSF для гнучкого розроблення ПЗ визначає п'ять Робочих Елементів для призначення та відстеження роботи. Загальна база даних Робочих Елементів та Схови-

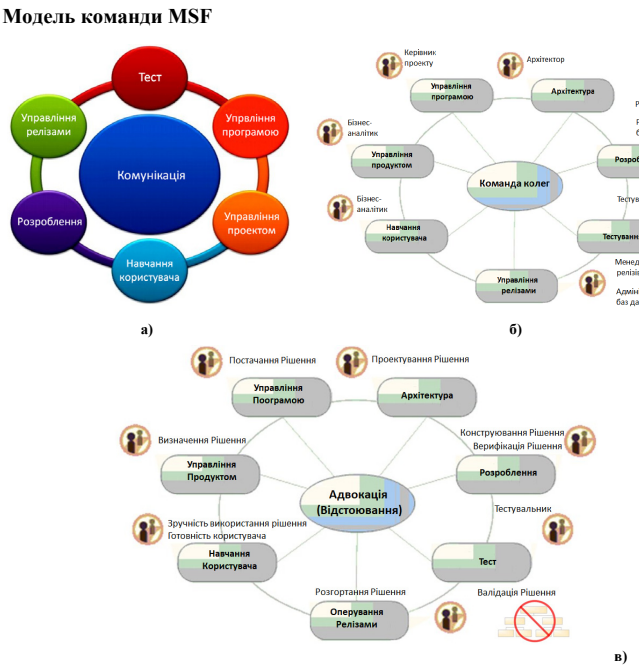


Рис. 8 – MSF модель команди: а) версія 2.0 [6], б) [8], в) версія 4.0 [7]

ще Метрик дозволяють відповідати на запитання щодо стану проекту в режимі реального часу.

Артефакти (Робочі Продукти)

Формулювання бачення, Архітектура системи, План тестування, Персони, Сценарії, Розкадровки, Завдання розроблення, Моделі інтерфейсів, Архітектурні прототипи, Модульні тести, Класи, Набори змін, Зареєстровані нотатки, Тестові сценарії, Звіти про помилки.

Робочі Продукти (Work Products) за [8]. Робочі Продукти – це документи, електронні таблиці, плани проектів, вихідний код та інші відчутні результати дій (діяльності).

Виконавці

Еволюція Моделі команди MSF показана на Рис. 8.

Групи підтримки: Архітектура, Розроблення, Управління продуктами, Управління програмами, Управління релізами, Тестування, Навчання користувача.

Хто що відстоює (advocacy): Архітектура – відстоює Систему в цілому, Розроблення – відстоює Технічне Рішення, Управління Продуктом – відстоює Бізнес Клієнта, Управління Програмою – відстоює Надання Рішень, Управління Релізами – відстоює Безперебійне Надання та Розгортання Рішення у Відповідній Інфраструктурі, Тестування – відстоює Якість Рішення з точки зору Клієнта, Навчання Користувача – відстоює Найефективніше Рішення в очах Цільових Користувачів.

Командні принципи

Команда колег з чіткою підзвітністю, спільною відповідальністю та відкритою комунікацією. Кожна роль відповідає за певну частку якості загального рішення.

Захист усіх ключових зацікавле-

них сторін, які потрібно представити в успішному проекті розроблення ПЗ. Кожна точка зору представлена для забезпечення механізмів контролю та противаг, що запобігають помилкам, упущенням та однобоким рішенням.

Розтягнути, щоб відповідати масштабу, необхідному для конкретного проекту. Складові можуть бути об'єднані в невеликі команди або додатково покращені, якщо команди масштабуються для більших проектів.

Хто що робить

Бізнес-аналітик: Фіксує бачення проекту, Створює вимоги до якості обслуговування, Створює сценарії.

Менеджер проекту: Керує проектом, Планує ітерації, Керує ітераціями.

Архітектор: Створює архітектуру рішення.

Розробник: Створює продукт, Виправляє помилки, Реалізує завдання розроблення.

Тестер: Закриває помилку, Тестує вимогу якості обслуговування, Тестує сценарії.

Менеджер випусків: Випускає продукт.

Висновки

У статті досить детально розглянута методологія розроблення ПЗ MSF Agile. З одного боку, показані співвідношення її конструкцій з метамоделлю зі стандарта [3]. З другого боку, показана їх відповідність з конструкціями відповідних КаРі.

Поєднання обох методологій – MSF і Agile – не є необхідним. Однак їх знання і розуміння зменшить кількість помилок в проектах розроблення ПЗ, а також дозволить вибирати практично економніший і ефективніший шлях їх виконання.

Автори вважають, що викладеного матеріалу достатньо, щоб зацікавлена компанія змогла б створити власну методологію, що базується на MSF Agile. При цьому використання Visual Studio Team System не є обов'язковим.

Важливою відмінністю статті є суміщений опис елементів методології і технології розроблення ПЗ. Цим вона суттєво відрізняється від багатьох джерел, в яких викладаються тільки окремі факти про методології (технології) розроблення ПЗ. Як правило, мало уваги звертається на ієрархію цих фактів, хоча вона є важливою. Очевидним прикладом є плутанина навіть у розумінні термінів методологія і технологія.

Список використаної літератури

1. Дишлик О., Чабанюк В. Каркасний підхід як стратегія дослідження проектування складних просторових інформаційних систем (на прикладі НІГД). *Землеустрій, кадастр і моніторинг земель.* 2025. № 1. С. 104–130. DOI: https://doi.org/10.31548/zemleustriy2025.01.09.
2. Чабанюк В., Дишлик О. Каркас рішень Microsoft (КаРі М) як узагальнена методологія каркасного підходу поводження з просторовими інформаційними системами. *Землеустрій, кадастр і моніторинг земель*. 2025. № 2. С. 88–105. DOI: https://doi.org/10.31548/zemleustriy2025.02.08.
3. Software engineering — metamodel for development methodologies : ISO/IEC 24744. 2nd ed. Geneva, 2014. 110 p.
4. Holt J. Systems Engineering Demystified: apply modern, model-based systems engineering techniques to build complex systems. 2nd ed. Birmingham : Packt Publishing, 2023. 504 p.
5. Brambilla M., Cabot J., Wimmer M. Model-

driven software engineering in practice. 2nd ed. Morgan & Claypool Publishers, 2017. 209 p. DOI: https://doi.org/10.2200/S00441ED1V01Y201208SWE001.
6. Wilson S. F., Maples B., Landgrave T. Analyzing requirements and defining solutions architecture. Redmond : Microsoft Press, 1999. 700 p.
7. Turner M. S. V. Microsoft Solutions Framework Essentials: building successful technology solutions. Redmond : Microsoft Press, 2006. 336 p.
8. MSF for Agile Software Development Process Guidance. Version 4.1.61114. 2024. URL: https://www.microsoft.com/en-us/download/details.aspx?id=5365 (дата звернення: 24.04.2026).
9. Дишлик О., Чабанюк В. Про методику каркасного підходу до створення просторових інформаційних систем з використанням сучасних технологій Microsoft. *Сучасні досягнення геодезичної науки та виробництва* : зб. наук. пр. Західного геодезичного товариства УТГК. Львів : Видавництво Львівської політехніки, 2026. Вип. 1 (51).
10. Pressman R. S., Maxim B. R. Software engineering: a practitioner's approach. 9th ed. New York : McGraw-Hill, 2019. 671 p.
11. Abrams S. Agile software development for beginners: mastering flexibility and efficiency in modern software projects. 2024. 114 p.
12. McConnell S. More effective agile: a roadmap for software leaders. Construx Press, 2019. 379 p. ISBN 978-1733518215.
13. Meier J. D. MSF Agile at a glance. 2005. URL: https://jdmeier.com/msf-agile-at-a-glance/ (дата звернення: 24.04.2026).
14. van Gigch J. P. System design: modeling and metamodeling. New York : Springer, 1991. 453 p.
15. Tekinerdogan B. Situational method engineering for constructing Internet of Things development methods. In: Business Modeling and Software Design. Cham : Springer, 2018. P. 221–239. DOI: https://doi.org/10.1007/978-3-319-94214-8_14.
16. Software development technologies : lecture course [Electronic resource] / V. O. Tykhokhod, A. L. Gurin, O. M. Bespala (comp.). Kyiv : Igor Sikorsky Kyiv Polytechnic Institute, 2024. 230 p. URL: https://agilemanifesto.org/iso/uk/manifesto.html (дата звернення: 24.04.2026).

References

1. Dyshlyk, O., & Chabaniuk, V. (2025). Karkasnyi pidkhid yak stratehiia doslidzhennia proektuvannia skladnykh prostorovykh informatsiinykh system (na prykladi NIHD) [Framework approach as a strategy for designing complex spatial information systems (case of NSDI)]. *Zemleustrii, kadastr i monitorynh zemel,* 1, 104–130. DOI: https://doi.org/10.31548/zemleustriy2025.01.09
2. Chabaniuk, V., & Dyshlyk, O. (2025). Karkas rishen Microsoft (KaRi M) yak uzahalnena metodolohiia karkasnoho pidkhodu povodzhennia z prostorovymy informatsiinymy systemamy [Microsoft Solutions Framework as a generalized methodology of the framework approach to spatial information systems]. *Zemleustrii, kadastr i monitorynh zemel,* 2, 88–105. DOI: https://doi.org/10.31548/zemleustriy2025.02.08
3. International Organization for Standardization. (2014). ISO/IEC 24744: Software engineering — Metamodel for development methodologies (2nd ed.).
4. Holt, J. (2023). Systems engineering demystified: Apply modern, model-based systems engineering techniques to build complex systems (2nd ed.). Packt Publishing.
5. Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-driven software engineering in practice (2nd ed.). Morgan & Claypool Publishers. DOI: https://doi.org/10.2200/S00441ED1V01Y201208SWE001

6. Wilson, S. F., Maples, B., & Landgrave, T. (1999). Analyzing requirements and defining solutions architecture. Microsoft Press.
7. Turner, M. S. V. (2006). Microsoft solutions framework essentials: Building successful technology solutions. Microsoft Press.
8. Microsoft. (2024). MSF for agile software development process guidance (Version 4.1.61114). Available at: https://www.microsoft.com/en-us/download/details.aspx?id=5365
9. Dyshlyk, O., & Chabaniuk, V. (2026). Pro metodyku karkasnoho pidkhodu do stvorennia prostorovykh informatsiinykh system z vykorystanniam suchasnykh tekhnolohii Microsoft [On the methodology of the framework approach to spatial information systems using modern Microsoft technologies]. In Suchasni dosiahnennia heodezychnoi nauky ta vyrobnytstva, 51. Lviv: Vydavnytstvo Lvivskoi politekhniky.
10. Pressman, R. S., & Maxim, B. R. (2019). Software engineering: A practitioner's approach (9th ed.). McGraw-Hill.
11. Abrams, S. (2024). Agile software development for beginners: Mastering flexibility and efficiency in modern software projects. SamIzdat.
12. McConnell, S. (2019). More effective agile: A roadmap for software leaders. Construx Press.
13. Meier, J. D. (2005). MSF agile at a glance. Available at: https://jdmeier.com/msf-agile-at-a-glance/
14. van Gigch, J. P. (1991). System design modeling and metamodeling. Springer.
15. Tekinerdogan, B. (2018). Situational method engineering for constructing Internet of Things development methods. In B. Shishkov (Ed.), Business modeling and software design (pp. 221–239). Springer. DOI: https://doi.org/10.1007/978-3-319-94214-8_14
16. Tykhokhod, V. O., Hurin, A. L., & Bespala, O. M. (Comp.). (2024). Software development technologies: Lecture course [Electronic resource]. Kyiv: Igor Sikorsky Kyiv Polytechnic Institute. Available at: https://agilemanifesto.org/iso/uk/manifesto.html.

.

Chabaniuk V., Dyshlyk O.

STANDARDIZATION OF MSF AGILE METHODOLOGY WITH THE USE OF ISO/IEC 24744

LAND MANAGEMENT, CADASTRE AND LAND MONITORING 2'26: 15-31.
http://dx.doi.org/10.31548/zemleustriy2025.02.02

Abstract. *The paper proposes the standardization of the software development methodology MSF Agile. It made by using the metamodel of software development methodologies from the ISO/IEC 24744 standard. Original of MSF Agile named MSF for Agile Software Development and denoted MSF4ASD. MSF4ASD implemented in MSF version 4.0 in 2005 p. in the tool/technology Visual Studio Team System. Therefore, here it is called an example of a specific software development methodology. Without implementation, the methodology exists, but it is better to call a specified generalized MSF methodology with the threat of losing practicality.*

Given the topicality of the Agile methodologies, MSF Agile can be a practically useful implementation of the generalized MSF methodology. The specific MSF Agile methodology can be obtained in two ways: 1) two-step specification or 2) standardization with subsequent specification. This article chooses the second way - first, MSF Agile standardized using the development methodologies metamodel from the ISO/IEC 24744 standard. After that, it is easier to perform practically

useful specification, since the task becomes typical. Implementation will be possible using various information technologies (IT), including Microsoft IT. The specific methodology can already be directly used in practice.

In the three main sections of this paper: 1) a standardization tool is introduced – the necessary elements of the software development methodologies metamodel from the ISO/IEC 24744 standard are described; 2) information about MSF4ASD is reminded, here simplified to MSF Agile; 3) MSF Agile is presented using the elements of the ISO/IEC 24744 metamodel. This shows how to make MSF Agile a standardized methodology. Then, by one-step reduction, it is possible to obtain a specific methodology that can constructively satisfy the Framework approach and facilitate the transition to Pattern-Based Spatial Engineering.

Keywords: *Microsoft Solutions Framework (MSF) Agile methodology, ISO/IEC 24744 MSF Agile standardization*