УДК 681.3

УДОСКОНАЛЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ СІМЕЙСТВ ПРОГРАМНИХ СИСТЕМ ЕЛЕМЕНТАМИ ГНУЧКИХ МЕТОДОЛОГІЙ

Г.І. Коваль, А.Л. Колесник, К.М. Лавріщева, О.О. Слабоспицька

Інститут програмних систем НАН України
03187, Київ, проспект Академіка Глушкова, 40, тел. (044) 526 4579

Процес створення сімейств програмних систем (ПС) за моделлю К. Пола вдосконалено практиками гнучких методологій розроблення ПС, спрямованими на критичне підвищення його гнучкості й ефективності співпраці учасників з нееквівалентними цілями. Інтегровано в ітеративному Процесі конструктивного аналізу ПрО. Запропоновано уніфіковані шаблони інженерії співпраці – у складі елементарної дії з співпраці та її типізованої форми (thinklet) – в ролі "блоків" для кроків раунду та опосередковано, для кроків долучених процесів: переговорів EasyWinWin і експертно-аналітичного оцінювання. Забезпечено переваги удосконаленого процесу для суб'єктів бізнесових і технічних цілей щодо створюваного сімейства ПС.

K.Pohl's Software Families engineering process is improved with Agile Practices aim at increasing of its agility and stakeholder collaboration effectiveness that is success-critical nowadays. These practices are integrated into multi-rounded Constructive Domain Analysis process. Unified Collaboration Engineering patterns that couple basic pattern of thinking and a parsimonious prescription for creating some variation on it (thinklet) – are proposed as building blocks for round steps and indirectly for the added processes steps both of the expert-analytical assessment and the negotiations EasyWinWin. The advantages of improved Software Families engineering process are achieved both for non-technicians and technical staff concerning to the Software Family being elaborated.

Вступ

Потреби сучасного ринку програмної продукції у різних предметних областях (ПрО) стимулюють організації-розробники (далі софтверні компанії) до швидкого створення та випуску на ринок високоякісних ПС і вчасного реагування на зміни вподобань замовників. Це, в свою чергу, ускладнює задачі, які покладаються на програмну інженерію, зокрема, пов'язані з вивченням умов ринку, попиту і контексту застосування розроблюваних ПС, залученням інвестицій, ефективним розподілом ресурсів, а також розробленням методологій виконання програмних проектів, придатних до адаптації відповідно до організаційних чинників (розміру команд проектів, рівня зрілості тощо).

Залежно від жорсткості і передбачуваності процесу розроблення ПС методології утворюють досить широкий спектр [1], на одному кінці якого – хаотичний недисциплінований процес, званий також «ковбойським» або «хакерським», а на іншому – повністю регламентований плановий процес у таких методологіях, як, наприклад, Cleanroom. Центральне місце у цьому спектрі займають гнучкі методології, насамперед, Scrum та XP (eXtreme Programming), які все ширше застосовуються у вітчизняних софтверних компаніях для розроблення одиничних ПС.

Цей спектр не охоплює методологій створення програмних систем у принципово іншій парадигмі – парадигмі сімейств ПС (СПС) у певній ПрО, де нові ПС генерується як члени сімейства з множини компонентів повторного використання (КПВ) в ході ретельно спланованого процесу інженерії СПС [2, 3].

Методології обох класів – гнучкого розроблення ПС (далі *AD-методологій*, від Agile Software Development) і розроблення ПС у парадигмі СПС (далі *СПС-методологій*) – спрямовані на досягнення трьох головних стратегічних цілей софтверних компаній: розширення ринкової сфери і збільшення обсягів випуску програмних продуктів високої якості; підвищення ефективності виробництва і зменшення витрат на їх створення; скорочення тривалості розробки. Але ці три цілі досягаються принципово різними способами, які надають розробникам розбіжні стратегії, методи і практичні прийоми (далі практики) проектування й реалізації ПС та керування проектами розробок. Наприклад, СПС-методології передбачають залучення значних інвестицій до формування еталонної архітектури сімейства продуктів з визначеним набором КПВ і встановленими межами її варіабельності для побудови архітектур окремих членів СПС, водночас як AD-методології пропонують легке інкрементне проектування архітектури кожної окремої ПС «з нуля», без використання КПВ. СПС-методології передбачають забезпечення якості ПС саме завдяки використанню перевірених КПВ і їх сполучень, тоді як AD-методології «ведуть» до якості ПС через використання особистого досвіду розробників, тісну комунікацію із замовниками, частий випуск версій продуктів на ринок (для скорочення часу виходу продукту на ринок) так званого time-to-market) і вчасне встановлення зворотного зв'язку для врахування швидких змін у вимогах замовників і безпосередніх користувачів ПС.

Обидва класи методологій мають як переваги, так і недоліки. Наприклад, очевидно, що у динамічних ПрО неможливо завчасно передбачити всі зміни, і це з часом позначиться на варіабельності СПС, трасовності її елементів (від вимог до архітектури і окремих її компонентів), а також на її цілісність, зокрема, через ускладнення процедур внесення змін до продуктів зі складу СПС, уже переданих споживачу. З іншого боку, в AD-методологіях відсутність, наприклад, процедури накопичення і використання КПВ негативно позначається на трудомісткості роботи розробників і обмежує накопичення колективного досвіду.

Збагачення СПС-методологій прийомами AD-методологій, зокрема, в частині комунікації із замовниками у процесах інженерії ПрО або керування змінами у СПС в ході виконання процесів інженерії ПС, сприятиме побудові більш гнучкої методології розроблення СПС, спрямованої на подальше зменшення витрат на розроблення і скорочення часу випуску ПС.

Мета роботи – вдосконалення традиційного процесу розроблення СПС елементами гнучких методологій для першочергового опрацювання його обмежень, найбільш критичних за умов слабко передбачуваних глобальних ризиків сучасних програмних проектів. Засіб її досягнення – інтеграція до процесу планування СПС: задач підвищення гнучкості та ефективності співпраці суб'єктів усіх нееквівалентних бізнесових і технічних цілей щодо СПС; принципів і практик розв'язання цих задач у гнучких методологіях; базових засобів Інженерії співпраці (Collaboration Engineering) [4] та процедур і методів процесу експертного оцінювання [5] для створення адекватної методичної підтримки обраних практик. Пропонований підхід розроблено при виконанні досліджень за проектом ІПС НАН України ІІІ-1-07 «Розробка теоретичного фундаменту генеруючого програмування та інструментальних засобів його підтримки».

1. Парадигма розроблення сімейства програмних систем

1.1. Характерні ознаки парадигми СПС. У задачі інженерії СПС входить використання знань у певній ПрО для побудови варіабельного сімейства продуктів, яке б задовольняло потреби множини замовників, що працюють у цій ПрО. Основними рисами функціонуючого СПС є формування і дотримання планів виробництва членів СПС і підтримання впорядкованої еволюції СПС, засноване на контролюванні сфери його поширення.

Характерними ознаками парадигми СПС, у тій чи іншій мірі підтриманими СПС-методологіями, є [6]:

– *наявність набору (лінійки) програмних систем*. Множина програмних продуктів або вбудованого програмного забезпечення систем розробляється для багатьох замовників за різними специфікаціями. Лінійка продуктів здатна покрити потреби одночасно багатьох замовників з різними потребами в одній ПрО. Через широке охоплення ринку досягається економія витрат;

– *використання бази готових ресурсів – активів ПрО*. База активів (або база компонентів) використовуються як основа для розроблення або конфігурування специфічних програмних продуктів;

– *виокремлення у процесі створення ПС двох синхронно виконуваних і керованих підпроцесів*, а саме: розроблення готових ресурсів КПВ та розроблення ПС з використанням готових ресурсів;

– *мислення в термінах загальних (незмінних, спільних для всіх ПС) і варіабельних характеристик СПС*. Мислення в термінах характеристик ПС, значущих для замовників, необхідне при визначенні економічних переваг, функційного покриття ПрО, обсягів супроводження СПС тощо;

– *стратегічне планування багаторазового використання готових ресурсів*. Передбачає стратегічне мислення в широкому масштабі на додаток до мислення в термінах характеристик СПС;

– *мислення в термінах властивостей ПрО для кращого розуміння передових практик, потреб і проблем у ній*. Потрібне на додаток до знань у ПрО для визначення спектра вирішуваних задач у ПрО, зацікавленості замовників у програмних продуктах, а також підходів до забезпечення якості ПС.

1.2. Моделі процесу розроблення ПС у парадигмі СПС. Загальний процес побудови і функціонування СПС ґрунтується на можливості повторного використовування вимог, архітектури та компонентів, поданих документами, моделями, кодом, тестами тощо у репозиторії СПС і забезпечених здатністю до змінювання, тобто *варіабельністю*. Варіабельність може бути реалізована як у СПС, так і в конкретних ПС. У першому випадку забезпечується варіабельність *виробничої технологічної лінії*, яка підтримує існування сімейства як множини ПС, а в другому – еволюційність ПС, «відлучених» від СПС. В обох випадках реалізація варіабельності сприяє забезпеченню життєздатності СПС і ПС [7].

Широко відомими є дві моделі подання діяльності з розроблення ПС у парадигмі СПС – модель Л. Нортроп (SEI) і модель К. Пола.

У моделі SEI (рис. 1) виділяються три види діяльності, які виконуються одночасно і доповнюють одна одну: *розроблення базових готових ресурсів* (ГОР), *розроблення ПС* та *керування* [8] .

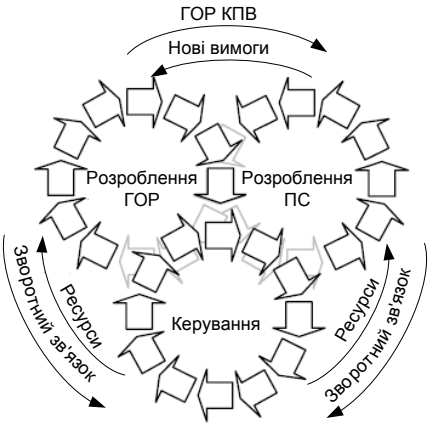

Рис. 1. Модель розроблення СПС SEI

Перший вид діяльності передбачає визначення сфери поширення СПС, розроблення ГОР, планування виробництва множини ПС з урахуванням контексту їх застосування (категорій користувачів, обов'язкових специфічних характеристик ПС тощо), обмежень щодо їх виробництва (зокрема, щодо застосовних платформ, особливостей архітектури, підходів до композиції ПС з ГОР) та стратегії виробництва (зокрема, стосовно залучення до створення СПС сторонніх організацій, визначення процедур замовляння-постачання ПС тощо). Діяльність з розроблення ПС включає побудову плану реалізації кожної ПС на основі визначеної сфери поширення СПС, множини розроблених ГОР та загального плану виробництва програмних систем на базі СПС, та наступне використання побудованого плану реалізації ПС для її інкрементного розроблення та постачання замовнику. Діяльність з керування зорієнтована за координування діяльності за вказаними напрямами і є дворівневою – передбачає вирішення завдань як організаційного, так і технічного керування проектами СПС та кожної ПС [3].

У моделі К. Пола виділяється множина процесів, виконуваних на двох рівнях – доменної інженерії (або інженерії ПрО), яку ще називають розробленням «для забезпечення повторного використання» (for reuse), і інженерії застосунків (або інженерії ПС), називаної також розробленням «із використанням КПВ» (with reuse) [9]. На відміну від моделі SEI, у цій моделі діяльність з керування не виділена явно, а є складовою процесів доменної інженерії СПС, і встановлюється зворотний зв'язок від процесів інженерії застосунків до процесів доменної інженерії (рис. 2).

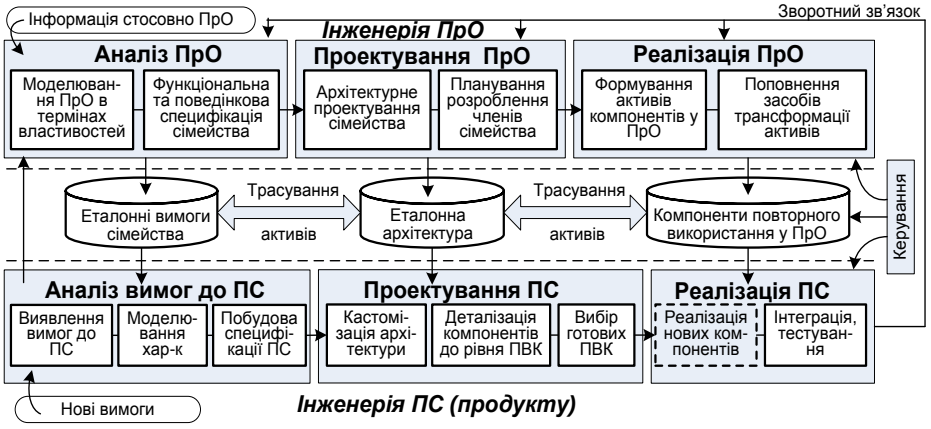

Рис. 2. Модель процесу розроблення СПС К. Пола

Сімейство ПС є досить новим об'єктом вітчизняної програмної інженерії, стосовно якого розробляються засади інженерії якості СПС [10, 11], а також керування проектами СПС та ПС [3].

Оскільки варіабельність є невід'ємною характеристикою сімейства ПС, а окремими задачами є моделювання та розроблення механізмів варіабельності [12], трасування властивостей сімейства на рівнях еталонних вимог, архітектури та компонентів, аналіз та відновлення цілісності сімейства після його доповнення новими артефактами, поява яких обумовлена долученням нового члена до сімейства тощо.

Відомими СПС-методологіями є: PuLSE (Product Line Software Engineering), SPLIT (Software Product Line Integrated Technology), FOPLE (Feature-Oriented Product Line Engineering), FeatuRSEB (Featured Reuse-Driven Software Engineering), PLUS (Product Line UML-based Software Engineering), PLUSS (Product Line Use case modeling for Systems and Software) тощо [13].

На додаток до названих, існує низка СПС-методологій, призначених для розроблення сімейств ПС за іншою моделлю – моделлю *сімейств процесів* [14]. Вони «розглядають» ПС через призму не множини виконуваних нею функцій, а множини процесів у ПрО, для автоматизації яких розробляється ПС, поданих сценаріями функціонування ПС у ПрО. Ці методології доцільно використовувати для розроблення інформаційних систем у сфері електронної комерції, систем керування потоками робіт, систем керування роботою технічних пристроїв тощо. На відміну від вищезгаданих методологій, вони передбачають накопичення і повторне використання фрагментів загальних та варіабельних *сценаріїв роботи* в ПрО.

1.3. Переваги використання парадигми СПС. Головні переваги інженерії СПС можна поділити за трьома категоріями: зниження витрат на розроблення, підвищення якості продукції і скорочення часу до випуску на ринок (або постачання замовнику).

Використання парадигми СПС сприяє кращому розумінню ПрО і перспектив її еволюції, полегшує накопичення досвіду і підготовку спеціалістів-аналітиків у ПрО. Повторне використання описів вимог до ПС обумовлює покращенню аналізу потреб клієнтів і підвищення їх довіри до майбутніх ПС.

Стратегія розроблення з КПВ сприяє зменшенню трудовитрат на кодування, супроводження і тестування ПС, а впровадження стандартів розробки (програмування, документування) – скороченню дублюючих надмірних реалізацій і, в перспективі (за умови повільної еволюції ПрО), повному повторному використанню існуючих ГОР. До цього можна додати полегшення обслуговування програмного забезпечення, керування змінами, підвищення контрольованості процесу долучення нових характеристик тощо. Аналіз імплементованих продуктів надає також переваги в оцінюванні вартості майбутніх.

Впровадження СПС дозволяє підвищити якість ПС. Використання бази спільних ГОР обумовлює уніфіковане внесення змін у всі ПС – члени сімейства. Оскільки окремі ГОР і архітектурна платформа СПС верифікуються до використання, створювані ПС забезпечуються «вбудованою» якістю перед конфігуруванням. Поліпшують якість також цикли зворотного зв'язку між інженерією застосунків та інженерією ПрО, що сприяють інкрементному вдосконаленню процесів у проектах ГОР, ПС і СПС [3].

Завдяки накопиченню і використанню КПВ зменшується тривалість розроблення кожної нової ПС і скорочується час подання ПС на ринок або замовнику, що сприяє підвищенню доходів софтверної компанії.

Перевагами парадигми СПС є також упорядкування довготривалих процесів, пов'язаних зі стратегічним плануванням, виробленням стратегії маркетингу, політики залучення інвестицій тощо. Сприяють підвищенню ефективності СПС чинники організаційної структури, культури, навчання тощо.

1.4. Обмеження використання парадигми СПС. Однією з головних проблем використання парадигми СПС є орієнтація на конкретні досить стабільні ПрО і кола замовників, для яких розробляються СПС. Хоча мають місце важливі для софтверної компанії інвестиції, але й зростають ризики і впливи на компанію та її керівництво, пов'язані з еволюцією менталітету, а також витратами часу і грошей. У цих умовах керівництво компанії має зосередитися на середньо строковій перспективі розгортання СПС, не очікуючи негайних результатів.

Зміна парадигми розроблення – з одиничних ПС на їх сімейства, що зумовлює явища опору до новацій через необхідність зміни організаційної будови і обов'язків співробітників. Зазвичай лише одиниці спеціалістів мають глобальне (масштабне) бачення архітектури СПС, серед яких має бути знайдений той, хто найкраще знає ПрО, має достатню відповідальність, повноваження і досвід у компанії, а також розуміння теорії побудови програмних систем у парадигмі СПС.

Сприйняття парадигми СПС обумовлює також потребу розповсюдження інформації по організації, особливо стосовно її архітектури та шляхів еволюціонування, які мають бути відомими кожному і вчасно. Проте у великих компаніях комунікація співробітників, залучених до процесів розроблення ГОР, СПС та ПС, ускладнена.

Деякі складнощі можуть виникнути при прийманні рішень щодо розроблення нового продукту в парадигмі СПС. Еволюція сфери охоплення ПрО вимагає обережного уточнення сферу поширення СПС. Надто широка сфера спричиняє суттєве ускладнення ГОР і потенційне зниження їх ефективності як КПВ, а надто звужена сфера поширення не виправдовуватиме витрат на розроблення і супроводження ГОР. Складним є розподіл і оцінювання ресурсів для запровадження та супроводження СПС.

Використання парадигми СПС передбачає наявність великих фреймворків і не може бути організоване швидко і без ризиків. Істотні зусилля мають бути прикладені для створення бізнес-цінностей СПС, які часто визначаються в ході формування сфери поширення СПС або процесу керування продуктом.

Значні зусилля і ризики пов'язують також з діяльністю щодо розроблення ГОР КПВ у проактивний (попереджувальний) спосіб у фазі доменної інженерії через вивчення й оцінювання майбутніх потреб ПрО. Ці зусилля зазвичай потребують залучення значних витратних ресурсів. ГОР можуть бути:

– розроблені в рамках проектів ГОР софтверної компанії, яка працює в парадигмі СПС;

– закуплені на ринку як комерційні COTS продукти (безпосередньо або через отримання ліцензійних прав на використання (open source компоненти або Web-сервіси));

– замовлені через відкриття проектів на розроблення спеціалізованих ГОР для СПС;

– виявлені (видобуті) в раніше створених ПС софтверною компанією (і успадкованих).

Для виконання аналізу можливостей використовуються методи якісного та кількісного аналізу рішень (decision analysis) в умовах невизначеності, множини конфліктуючих цілей або динамічних змін з урахуванням бізнесових, технічних або політичних чинників щодо вартості, план-графіка, штату, очікуваної якості, архітектури, супроводжуваності ГОР тощо, а також стратегії виробництва та виробничих обмежень СПС. Аналіз рішень може засновуватися на застосуванні байєсівських мереж, які поєднують чинники, пов'язані причинно-наслідковими або умовними залежностями.

Проектування архітектури СПС ускладнене через відсутність настанов, методів і інструментів для подання або верифікації еталонних архітектур й інтегрованих інструментальних засобів (середовищ) розроблення і експлуатації сімейств ПС.

Однією з найважливіших проблем у функціонуванні СПС є керування варіабельністю [3]. Значних зусиль потребує формування ясного бачення того, які варіації потрібні, як вони мають бути описані, і якими механізмами реалізовані [12]. По мірі еволюції СПС керування варіабельністю може ускладнитися через зростання кількості варіацій і ускладнення взаємозв'язків та залежностей між ГОР, наділених варіантами.

Існують також проблеми з актуалізацією не програмних ГОР (моделей, документів) у репозиторії СПС, кількість і складність яких з часом зростає через поповнення множини спільних і варіабельних артефактів. Ускладнюються процеси керування змінами як програмних, так і не програмних ГОР, а також керування конфігурацією СПС і, відповідно, вартість супроводження СПС.

2. Парадигма гнучкого (agile) розроблення програмних систем

2.1. Характерні ознаки парадигми AD. Гнучке розроблення ПС як парадигма еволюціонувала від швидкого розроблення застосунків (RAD, від Rapid Application Development) і ранніх спіральних моделей до положень маніфесту "Manifesto for Agile Software Development" (http://agilemanifesto.org/), який фіксує основні ознаки парадигми AD:

– *найвищий пріоритет – задоволення вимог замовника* шляхом швидкого і безперервного постачання найбільш цінного для замовника працездатного коду. Працездатний код є основною мірою прогресу;

– *розроблення застосунків невеликими, але працюючими порціями*. Працездатний код поставляється якомога частіше, періодами від пари тижнів до пари місяців.

– *швидке реагування на зміни*. Вітаються змінювані вимоги, навіть на пізніх фазах розроблення. Зміни розглядаються як стимул підвищення конкурентоспроможності продукту;

– *динамічний ітеративний життєвий цикл, орієнтований на попередження ризиків.* Ітеративність з короткими циклами дозволяє виконувати швидку верифікацію і реалізацію, а також мінімізувати ризик;

– *стиль роботи: тісна взаємодія і співпраця*. Замовники і розробники впродовж всього проекту щодня працюють разом. Встановлюється зручний режим ведення розробки, який дотримується спонсорами, розробниками і користувачами в ході виконання проекту;

– *спосіб комунікації – безпосереднє спілкування*. Найбільш ефективним і дієвим способом передачі інформації (як усередині команди розробників, так і зовні) є розмова «лицем до лиця»;

– *орієнтація на людський чинник*. Проекти вибудовуються навколо вмотивованих особистостей, щодо яких виказують довіру і яким створюють всі необхідні умови роботи;

– *самоорганізація команди*. Кращі архітектурні рішення, набори вимог і дизайн створюються самоорганізованими командами;

– *постійне навчання*. Команда впроваджує будь-які методи підвищення власної ефективності.

– *мінімізація дій в процесах*. Простота – це мистецтво максимізації кількості невиконуваної роботи. Мінімізація дорогих проміжних артефактів проекту та внутрішньої документації;

– *орієнтація на постійне вдосконалення коду*. Дотримання прийнятих стандартів кодування, підтримка технологій рефакторінгу, безперервне здійснення процесу проектування, безперервне підвищення технічної досконалості, покращення дизайну, автоматичне тестування, неперервна інтеграція коду;

– *формулювання системних метафор*, на основі яких формується і в термінах яких в подальшому уточнюється високорівнева схема проекту. Вибір метафори – один з найскладніших, але ефективних процесів для встановлення взаєморозуміння із замовником на етапі проектування.

У цій парадигмі розроблено чимало AD-методологій, зокрема, DSDM (Dynamic Systems Development Method), AgileRUP, LSD (Lean Software Development), Scrum, XP (eXtreme Programming), FDD (Feature Driven Development), Crystal [6], Adaptive Software Development тощо [1, 15]. До 2005 року найбільшою популярністю користувалась методологія XP, а після 2005 – Scrum, гібридна методологія «XP+Scrum» [15].

Більшості AD-методологій притаманна низка спільних практик – ітеративне розроблення через тестування (test-driven development), парне програмування (peer programming), колективне володіння кодом, гра в планування (planning game), гнучке моделювання, використовування CRC-карток (від Class, Responsibilities and Collaboration), пробних рішень для зменшення ризику, технологічних прототипів тощо [15]. Ці та інші практики формують складний і тісно взаємозв'язаний фреймворк, який дозволяє значно мінімізувати обсяг ризиків проекту ПС.

2.2. Модель процесу розроблення ПС у парадигмі AD. Узагальнена модель процесу розроблення в AD-методологіях показана на рис. 3.

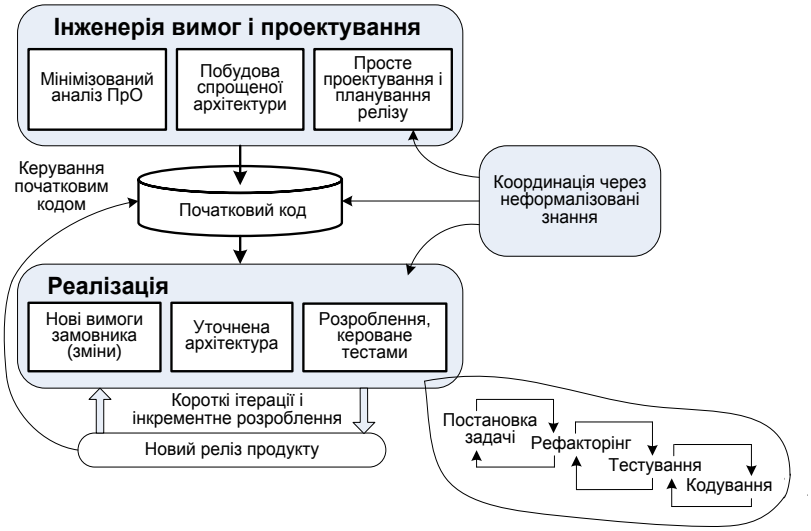

Рис. 3. Загальна модель процесу розроблення в ASD-методологіях

У цій моделі мінімізовані практики інженерії вимог і моделювання архітектури з метою якнайшвидшого переходу до процесу кодування першої версії продукту. Команда проекту застосовує інкрементне розроблення з короткими ітераціями.

Кінцевим результатом етапу планування є перелік завдань, що підлягають реалізації на наступній ітерації. Кожний розробник (або пара розробників) отримує завдання, бере відповідний існуючий фрагмент коду, виконує його рефакторинг, необхідний для спрощення написаного коду, складає тести і лише потім створює сам код, який піддаватиметься тестуванню. Розробник «перебуває» в циклі «тест-код» доти, поки завдання не буде завершене повністю (а розроблений код повністю протестований). Після цього робиться спроба переробити програму, щоб спростити код. Перероблення програми починається з внесення змін в дизайн. Оскільки цикли «дизайн–тест–код» нетривалі, а замовник часто одержує працюючі версії програмного продукту, зворотний зв'язок здійснюється безперервно і служить для контролю того, що проектування і кодування просуваються в потрібному напрямі. Оскільки зміни на кожному циклі малі, рішення, від яких доводиться відмовлятися, невеликі, внаслідок чого відбувається швидка реакція на зміни з якнайменшими витратами.

Створювані розробниками артефакти піддаються колективному контролю і керуванню всіма членами команди. Початковий код розміщується в репозиторії і перебуває в сфері процесу керування конфігурацією. Перебіг процесу розроблення залежить від фреймворка обраної AD-методології та динамічно коригується з огляду на розуміння ситуації (неформалізовані знання) менеджерів команди, які здійснюють координацію зусиль у проекті. Але будь-яка з методологій передбачає неперервну комунікацію у проекті, тісну взаємодію з замовником, постійне оновлення коду в репозиторії через його рефакторінг і тестування.

2.3. Переваги використання парадигми agile. Головна перевага AD-методологій – здатність софтверної компанії працювати в нестабільній ПрО, своєчасно реагуючи на часті і несподівані зміни у вимогах і потребах замовника впродовж всього процесу розроблення. Керування змінами вимог грунтується на ітераційному розробленні, завдяки якому вимоги аналізуються й уточнюються на початку кожного циклу. Таким чином, вимоги, які є гнучкими і змінюваними в часі, більше відповідатимуть здогадним потребам замовника, а безперервна інтеграція сприятиме підвищенню якості продукту.

Наступна перевага AD-методології – зменшення ризиків проекту і підвищення його ефективності через тісну взаємодію із замовником, безперервне тестування і прозорі комунікації. Безперервне інтегрування і тестування продукту впродовж життєвого циклу сприяють зниженню інтенсивності дефектів і ранній ідентифікації проблем. Самоорганізація команд зумовлює спроби поліпшити комунікацію в межах команди, що сприяє скороченню тривалості процесу приймання рішень. Інкрементне розроблення і формування спільного для замовника і розробника бачення того, що має розроблятися, допомагає зосередитися на найбільш вагомих задачах.

Ще одна перевага, яка досягається завдяки ітераційному інкрементному процесу розроблення, декларованому всіма AD-методологіями, – скорочення часу до випуску продукту. Загальним для методологій є наголос на постачанні працюючого коду, отриманого в кожній ітерації, хоча методології відрізняються встановленою тривалістю ітерацій, складом і послідовністю вирішуваних у них задач [15]. Теоретично, переданий код міг би одразу використовуватися замовником без очікування завершення проекту.

Загальна перевага всіх AD-методологій – задоволеність замовників, в основі якої лежить щоденна взаємодія з ними розробників. Контролюючи еволюцію змін вимог до продукту, розробники навчаються точно інтерпретувати вимоги і точніше реалізовувати їх у коді. Інший аспект задоволеності замовників –орієнтація на першочергову реалізацію найбільш значущих (цінних) для замовників функцій продукту, заснована на переоцінюванні пріоритетів проекту в кінці кожної ітерації.

Численні переваги надають AD-методологіям використовувані практики (табл. 1).

Таблиця 1. Переваги практик, застосовних в AD-методологіях

Практика	Переваги застосування в AD-методологіях
Парне програмування	Підвищення дисципліни розробників, якості коду; вільне поширення вербальної інформації.
Гра в планування	Швидке формування приблизного плану роботи і постійне його оновлення в міру того, як умови задачі стають все більш чіткими. Замовник відповідає за ухвалення всіх бізнес-рішень, тоді як команда ухвалює всі технічні рішення. Планування виконується в три етапи: початкове планування, планування випуску версії продукту і планування ітерації
Складання тестів	Переваги тестів як форми документації: не містять ніяких зайвих відомостей; дають уявлення про очікувані сценарій поведінки коду; завжди в точності відповідають поточному стану коду
Часта інтеграція коду	Переваги інтеграції коду кілька разів на день: зниження ризику не встигнути завершити роботу в заданий термін; спрощення локалізації й виправлення дефектів (у доданому коді); доступ усіх членів команди до самого останнього повністю працездатного коду
Колективне володіння кодом	Дотримання стандартів кодування; відповідальність кожного за коректне функціонування коду, який він розробляє або змінює; зменшення залежності команди від кожного окремого її члена.
Рефакторинг	Покращення структури коду, спрощення його розуміння, модернізації, оптимізації; видалення дублюючого коду; як засіб пошуку дефектів
Документування	Поділ документації на зовнішню (для замовника) і внутрішню (для розробників). Зовнішня паперова документація (документи) складається наприкінці розроблення для стабільного продукту. Внутрішню документацію складають всі артефакти проекту (код, тести, CRC-картки)

2.4. Обмеження використання парадигми agile. Впровадження AD-методологій у поточний процес програмної інженерії компанії спричинює організаційні проблеми, як наприклад: необхідність реформування організаційної культури (перехід до роботи в командах); залучення замовника до процесу розроблення (запровадження нових правил комунікації, переосмислення ролі замовника); спротив персоналу до змін (перехід від документо-орієнтованого розроблення до гнучкого, необхідність оновлення знань); складність розподілу ресурсів і встановлення економічного балансу між інвестиціями і прибутком у проекті.

Оскільки вимоги в AD-методологіях є «полегшеними» й керованими локально в межах кожної ітерації, можуть з'явитись проблеми щодо їх стабілізації, правильного тлумачення. Можливі також проблеми, пов'язані з уривчастістю архітектури, ускладненням рефакторингу тощо.

Загалом використання AD-методологій не рекомендоване для виконання крупних проектів (з чисельністю персоналу більше 40 осіб) через труднощі організації безпосередньої комунікації членів команди «лицем до лиця». Ці методології не застосовні також для розроблення ПС, до яких висуваються високі вимоги надійності і безпеки.

3. Підхід до вдосконалення процесу розроблення СПС

3.1. Сутність та основні конструкти підходу. Аналіз обмежень традиційних методологій розроблення СПС визначає дві вимоги їх опрацювання, що стають першочерговими за умов слабко передбачуваних багатоаспектних ризиків сучасних програмних проектів. Першою є підвищення ефективності співпраці *стейкхолдерів* – суб'єктів нееквівалентних "бізнесових" цілей експлуатації елементів СПС та "технічних" цілей їх розробки й повторного використання в ролі ГОР. Друга – це забезпечення чутливості процесу розроблення СПС до неочікуваних змін як ділового середовища компанії, так і поточних артефактів співпраці стейкхолдерів.

У свою чергу, огляд AD-методологій засвідчує їх зорієнтованість саме на задоволення цих вимог. Але водночас він висвітлює необхідність адекватної методичної підтримки забезпечуючих AD-практик (див. табл. 1) для розкриття їх потенціалу за умов багаторазового спадкоємного застосування в руслі парадигми СПС.

Створення середовища такої підтримки та розгортання в ньому AD-практик за AD-принципами вже на першому етапі процесу інженерії ПрО і складає сутність пропонованого підходу до вдосконалення процесу розроблення СПС. Для цього універсальні засоби Інженерії співпраці (Collaborative Engineering) [4] доповнено процесом обгрунтованого експертно-аналітичного оцінювання, що запропонований авторами в [5].

Основними теоретичними конструкціями підходу є:

а) прийнята модель процесу розроблення СПС;

б) модель процесу інженерії ПрО, вдосконаленого AD-практиками й середовищем їх застосування;

в) засоби методичної підтримки вдосконаленого процесу згідно вимог а), б).

За результатами аналізу традиційних методологій процесу розроблення СПС [8–11, 13, 14] (див. підрозділ 1.2), для опрацювання обрано модель К. Пола. Згідно з нею, "мішенню" вдосконалення є процес аналізу ПрО із своїм інформаційним середовищем (див. рис. 2).

Пропонований результат вдосконалення – процес конструктивного аналізу ПрО. Він визначений як послідовність уніфікованих раундів формування / актуалізації узгоджених версій концептуальної моделі СПС (так званої Карти продуктів) [9] та Плану створення СПС, спадкоємно виконуваних в інформаційному середовищі, що передбачене моделлю Пола. Карта продуктів відображає еталонні вимоги до СПС – загальноприйнятний для всіх стейкхолдерів склад охоплених сегментів ПрО, пріоритезованих програмних продуктів та функціональних властивостей для кожного з них. Вона надає підґрунтя комунікації стейкхолдерів у наступних раундах, а також учасників подальших етапів процесів інженерії ПрО та інженерії застосувань.

Модель процесу конструктивного аналізу ПрО являє собою трійку

$$CDA = \langle ST, EN, RM \rangle, \qquad (1)$$

де *ST*, *EN* і *RM* – склад стейкхолдерів, інформаційне середовище та підмодель окремого раунду процесу.

Склад стейкхолдерів *ST* та їх функції в процесі конструктивного аналізу ПрО подано в табл. 2.

Таблиця 2. Склад стейкхолдерів процесу конструктивного аналізу ПрО

Стейкхолдер	Ролі в процесі конструктивного аналізу ПрО
Суб'єкти бізнесових цілей	
Представник вищого керівництва	Репрезентує стратегічне бачення діяльності компанії. Розподіляє час та ресурси. Обстоює цілі успішного, ресурсно-ефективного, вчасного, стабільного створення СПС з мінімальним ризиком. Забезпечує досягнення решти бізнесових інтересів компанії щодо СПС
Споживач	Надає незаперечне судження щодо продуктів та їх властивостей
Суб'єкт маркетінгу та продажу	Надає ознайомлювальну інформацію щодо потреб споживачів та можливостей конкурентів і їхніх продуктів. Подає оцінки термінів та обсягів продажів
Суб'єкти технічних цілей	
Архітектор	Надає деталізовану технічну інформацію щодо наявних продуктів та оцінки впливу рішень стосовно Карти продукту на еталонну архітектуру
Розробник	Оцінює альтернативні варіанти Карти продукту з позицій витрат на повторне використання ГОР для їх реалізації
Фахівець з підтримки	Надає ознайомлювальну інформацію щодо переваг і недоліків існуючих продуктів. Оцінює варіанти актуалізації Карти продукту за критеріями легкості надання продукту споживачам та забезпечення сумісності з існуючими продуктами

У свою чергу, середовище *EN* з моделі (1) поєднує шість банків даних (БД):

– БД, передбачені моделлю К. Пола – еталонних вимог, еталонних архітектур та ГОР у ПрО;

– БД фактичної інформації щодо процесу розроблення СПС, поповнюваний у міру його перебігу;

– БД ретроспективи процесу експертного оцінювання, елементами якого є ієрархічні протоколи розв'язання задач оцінювання в раундах, запропоновані в роботі [5];

– БД ретроспективи процесу полегшених переговорів (EasyWinWin) [16] з арсеналу Інженерії співробітництва, утворений протоколами переговорів в раундах, визначеними в роботі [16].

У міру виконання раундів, БД еталонних вимог постійно поповнюється версіями Карти продуктів, а БД ретроспективи – відповідними протоколами.

Таким чином, завдяки уніфікації раундів у моделі *RM* та забезпечення їх спільним інформаційним середовищем *EN*, модель (1) надає усім стейкхолдерам *ST* поточного раунду можливість використання ретроспективи Карт продуктів, а також результатів переговорів і оцінювань для всіх попередніх раундів в повному обсязі разом із збереженням результатів цього раунду для подальшого використання.

3.2. Модель раунду конструктивного аналізу ПрО. Раунд складається з п'яти кроків. Їх склад та позицію у межах "рамкової" моделі Пола показано на рис. 4.

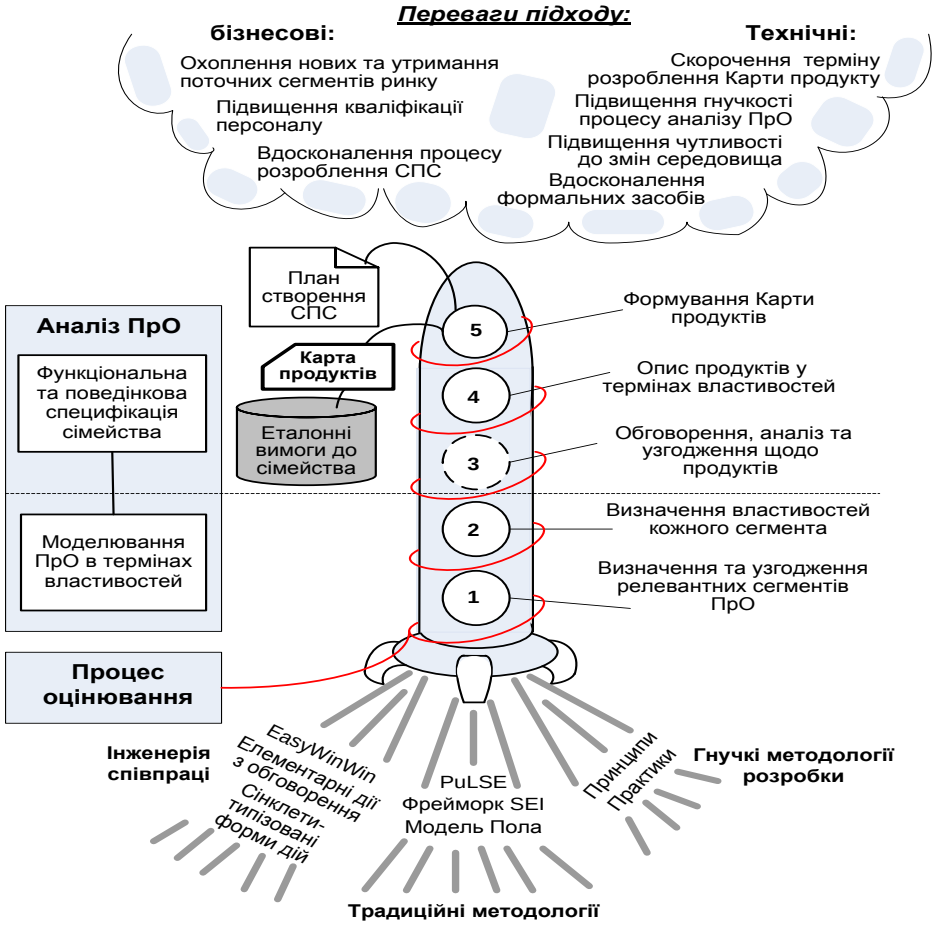

Рис. 4. Підгрунтя та структура раунду процесу конструктивного аналізу ПрО

Крок 1. Залучені стейкхолдери спільно обговорюють поточний перелік охоплених сегментів ПрО, досягаючи згоди стосовно їх складу та описів, а також долучення нових сегментів ПрО. Результатом кроку є остаточний перелік взаємоприйнятних описів сегментів ПрО.

Крок 2. Стейкхолдери здійснюють мозковий штурм з метою виявлення нових функціональних властивостей програмних продуктів у попередньо визначених сегментах ПрО. Потім вони надають індивідуальні оцінки ступеню притаманності кожної властивості з об'єднаного переліку кожному з цих сегментів. Нарешті, розгортається спільне обговорення з метою консолідації описів властивостей й усунення їх можливого дублювання, а також взаємоприйнятного зіставлення остаточних описів властивостей сегментам ПрО. При цьому кожна властивість може зіставлятися тільки одному сегменту. Взаємоприйнятна множина переліків властивостей для сегментів складає результат кроку.

Крок 3. До інформаційного контексту кроку належить бізнес-план компанії, результати аналізу потреб ринку і можливостей конкурентів (SWOT і PEST-аналіз) та початковий опис запитаних програмних продуктів. У процесі спільного обговорення цього опису в наведеному контексті стейкхолдери (насамперед споживачі) подають свої судження щодо бажаних характеристик продуктів. Непротирічне поєднання цих суджень з усуненням дублювання відіграє роль початкової версії метафори продуктів для наступного використання решти AD-практик з табл. 1. Уточнення й узгодження цієї версії між стейкхолдерами являє собою результат кроку.

Крок 4. Стейкхолдери подають індивідуальні оцінки ступеню притаманності програмним продуктам визначених властивостей. При цьому локалізуються неузгодженості оцінок та висвітлюються їх причини. Для опрацювання причин в разі необхідності повторно виконуються відповідні кроки з числа розглянутих. Нарешті, розгортається спільне обговорення для досягнення консенсусу щодо складу елементів Карти продуктів.

Крок 5. Визначені продукти, а потім і властивості, ранжуються за критеріями технічної здійснюваності й споживацької цінності. Аналогічно Кроку 4, виконується виявлення й локалізація неузгодженостей, опрацювання їх причин та переговорний пошук консенсусу щодо пріоритетів продуктів і властивостей для кожного продукту в Карті продуктів.

За моделлю Пола припустимі три форми виконання раунду в середовищі *EN* згідно моделі *RM*:

а) "реактивна", без Кроку 3 (особливий статус якого в раунді *RM* позначено на рис. 4 штриховою лінією) – при виявленні нових вимог до властивостей наявних або нових продуктів у охоплених сегментах ПрО;

б) "проактивна", без двох перших кроків, – при просуванні нових продуктів до нових сегментів ПрО;

в) "регламентна" з послідовним виконанням всіх п'яти кроків – при плановій актуалізації інформаційного середовища процесу розроблення СПС.

3.3. Засоби підтримки конструктивного аналізу ПрО. Для конструктивної реалізації AD-практик в раунді *RM* з (1) пропонується "конструювання" його Кроків 1-5 з уніфікованих "блоків". Роль останніх надано уніфікованим шаблонам ефективної організації елементарних переговорних дій з колективного досягнення певної мети [4]. Такий шаблон – це пара у складі елементарної переговорної дії (Породження, Обмеження, Впорядкування, Оцінювання, Формування консенсусу) та типізованої уніфікованої форми її здійснення, названої сінклетом (thinklet) – з числа сімдесяти, розроблених наразі в Інженерії співпраці [4]. Для повного розкриття потенціалу AD-практик безпосереднє використання шаблонів поєднано з опосередкованим – у складі кроків процесів полегшених переговорів EasyWinWin [16] та експертно-аналітичного оцінювання [5]. Внутрішня структура цих процесів та спеціальні методи індивідуального подання й інтелектуального узагальнення обґрунтованих експертних суджень, запропоновані авторами в [5], сприяють підвищенню обґрунтованості рішень, зменшенню ресурсних витрат на їх вироблення та забезпеченню інформаційної спадкоємності раундів конструктивного аналізу ПрО.

Призначення та взаємозв'язки пропонованих шаблонів на Кроках 1-5 раунду подано на рис. 5. Згідно з ним, поряд із безпосереднім використанням всіх шаблонів на Кроках 1-5, процес оцінювання підтримано шаблонами для дії "Оцінювання", а процес EasyWinWin – рештою шаблонів для всіх дій крім "Породження".

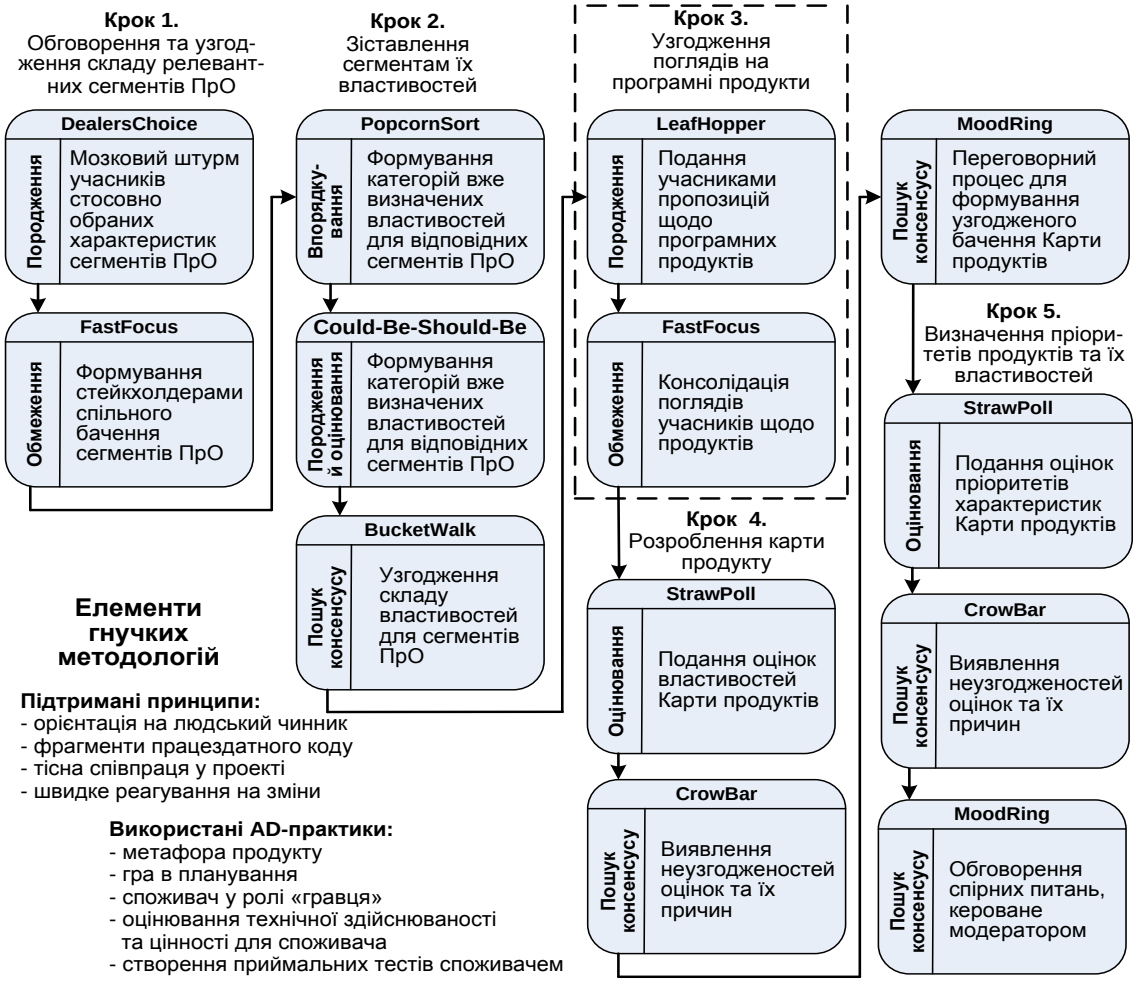

Рис. 5. Взаємозв'язки шаблонів співпраці в раунді конструктивного аналізу ПрО

Висновки

Проаналізовано переваги й обмеження гнучких методологій створення ПС та традиційних методологій розроблення сімейств ПС. Показано, що за умов слабко передбачуваних ризиків процесу розроблення сімейств ПС його вдосконалення вимагає насамперед підвищення гнучкості й ефективності співпраці суб'єктів технічних цілей створення елементів сімейства і бізнесових – їх експлуатації. Обрано принципи та практики гнучких методологій, перспективні для забезпечення цих вимог у середовищі адекватної методичної підтримки.

Традиційний процес створення сімейств ПС за моделлю К. Пола вдосконалено обраними практиками (метафора, гра в планування, залучення споживача до гри і побудови приймальних тестів, оцінювання вимог за технічною здійснюваністю та споживацькою цінністю). Вони інтегровані за принципами гнучких методологій вже на першому етапі процесу розроблення у вигляді ітеративного Процесу конструктивного аналізу ПрО.

Кроками його раунду є: узгодження складу релевантних сегментів ПрО та їх властивостей або досягнення згоди щодо майбутніх програмних продуктів; їх опис у термінах властивостей; формування Карти продуктів з їх пріоритетами і пріоритетами властивостей для продуктів за критеріями технічної здійснюваності та споживацької цінності. Запропоновано уніфіковані шаблони елементарних дій з ефективної співпраці із спільною метою (з арсеналу Інженерії співпраці) у ролі "блоків": безпосередньо для кроків раунду та, опосередковано, для кроків процесів полегшених переговорів EasyWinWin і експертно-аналітичного оцінювання, долучених з метою розкриття потенціалу гнучких практик із вдосконалення.

Запропоновані структура та засоби підтримки конструктивного аналізу ПрО забезпечують переваги вдосконаленого процесу розроблення. Для суб'єктів бізнесових цілей (вищого керівництва, промоутерів та споживачів) – це ресурсно прийнятне утримання поточних і охоплення нових сегментів ринку, підвищення кваліфікації персоналу, покращення іміджу створюваних продуктів і самої компанії, скорочення терміну надання продукту споживачам. "Технічними" перевагами – для архітекторів, розробників, персоналу супроводу – є забезпечення адаптивності процесу до непередбачуваних змін ділового середовища компанії, відсутність надлишкових прескриптивних обмежень, постійне підвищення інформованості учасників, підвищення обґрунтованості рішень, скорочення терміну і зниження витрат ресурсів на створення Карти продуктів.

Завдяки підвищеній гнучкості, вдосконалений процес є перспективним для розгортання як за первинного запровадження парадигми сімейства ПС у софтверній компанії, так і за умов переходу до цієї парадигми від методологій розробки окремих ПС. Його подальше вдосконалення передбачає побудову типових моделей оцінюваних характеристик (зокрема, у форматі Дерева цінності і мережі Байєса) та долучення комунікативного Дельфі-процесу для додаткової підтримки відповідних дій на кроках раунду.

1. *Основы* инженерии качества программных систем / Ф.И.Андон, Г.И.Коваль, Т.М. Коротун, Е.М.Лаврищева, В.Ю. Суслов // 2-е изд. – К.: Академпериодика. – 2007. – 672 с.
2. *Лавріщева К.М.* Генерувальне програмування програмних систем і їх сімейств // Проблеми програмування. – 2009. – № 1. – С. 3 – 16
3. *Лавріщева К.М., Коваль Г.І., Слабоспицька О.О., Колесник А.Л.* Особливості процесів керування при створенні сімейств програмних систем // Проблеми програмування. – 2009. – № 3. – С. 40 – 48.
4. *De Vreede G. J., Kolfschoten G.L.*, Briggs R.O. ThinkLets: a collaboration engineering pattern language // International Journal of Computer Applications in Technology. – 2006. – Vol. 25. – P. 140-154
5. *Лаврищева Е.М., Слабоспицкая О.А.* Подход к экспертному оцениванию в программной инженерии. – Кибернетика и системный анализ. – 2009. – № 4. – С. 151–168.
6. *Gylterud S.* Practices of Agile Software Product-Line Engineering. A qualitative assessment of empirical studies. – http://daim.idi.ntnu.no/masteroppgaver/IME/IDI/2007/3371/masteroppgave.pdf
7. *Ігнатенко П.П., Бистров В.М.* Особливості забезпечення життєздатності програмних систем в умовах генеруючого програмування // Проблеми програмування (Спецвипуск конференції УкрПРОГ-2008).– 2008. – № 2–3. – С. 270 – 278.
8. *Northrop L.M.* et al. Framework for Software Product Line Practice, version 5 // SEI. – 2007 – http://www.sei.cmu.edu/ productlines/ index.html
9. *Pohl K., Böckle G., Linden F.J.* Software Product Line Engineering: Foundations, Principles and Techniques. New York: Springer-Verlag. – 2005. – 437 p.
10. *Лавріщева К.М., Коваль Г.І., Коротун Т.М.* Підходи інженерії якості сімейств програмних систем // Проблеми програмування (Спецвипуск конференції УкрПРОГ-2008). – 2008. – № 2–3. – С. 219 – 228.
11. *Коваль Г.І.* Підхід до моделювання якості сімейств програмних систем // Проблеми програмування. – 2009. – № 4. – С. 49 – 58.
12. *Колесник А.Л.* Механізми забезпечення варіабельності в сімействах програмних систем // Проблеми програмування. – 2010. – № 1. – С. 35 – 44.
13. *Bayer J.* et al. PuLSE: a methodology to develop software product lines – http://www.cs.helsinki.fi/u/vjkuusel/ semma/1999.SSR_LosAngeles.pdf
14. *Bayer J., Buhl W., Giese C.* et al. Process Family Engineering. PESOA Report No. 18/2005, August 2005. – http://www.pesoa.org/pages/ Publications/Fachberichte062005/PESOA_TR_18-2005.pdf
15. *Шопырин Д.Г.* Управление проектами разработки ПО. Дисциплина «Гибкие технологии разработки программного обеспечения». – http://books.ifmo.ru/book/pdf/422.pdf
16. *Grunbacher P., Hofer C.* Complementing XP with requirements negotiation. In: Proceedings of the 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering, Agile Alliance, p. 105 – 108. – http://www.agilealliance.org/system/ article/file/909/file.pdf