УДК 519.164:004.14

П.І. Андон, О.О. Слабоспицька

ЗБІРКОВЕ ПРОГРАМУВАННЯ КОМПОНЕНТНИХ І СЕРВІС-ОРІЄНТОВАНИХ ПРИКЛАДНИХ ПРОГРАМНИХ СИСТЕМ

Зафіксовано основні проблеми розроблення сучасних прикладних застосунків. Проаналізовано особливості їх постановок у парадигмах компонентного та сервісного програмування. Підсумовано результати фахівців ІПС НАНУ з опрацювання цих проблем у парадигмах програмування збіркового типу (поняття й реалізація інтерфейсу, об'єктно-компонентний метод програмування, компонентні алгебри, методи перетворення типів даних, формальні засоби забезпечення змінності компонентних застосунків за рахунок управління їх варіабельністю). Показано розвиток доробку компонентного програмування в парадигмі сервісного програмування в семантичному Веб-середовищі (формальний апарат підтримки життєвого циклу композитного семантичного сервісу, методи оптимізації ярусно-паралельних композицій, моделі та методи забезпечення його адаптивності). Окреслено засоби технологічної підтримки застосування розробленого формального апарату компонентного програмування. Визначено проблеми для подальшого опрацювання в парадигмі сервісного програмування.

Ключові слова: збіркове програмування, сервісне програмування, метод зборки, інтерфейс, семантичний Веб-сервіс, компонування семантичних Веб-сервісів, об'єкт, компонент, інженерія якості, стандартизація програмного забезпечення, ярусно-паралельний граф, адаптивний композитний сервіс.

Актуальні проблеми та засади їх опрацювання

За умов стрімкого розвитку програмної індустрії у світі та в Україні дедалі перспективнішою парадигмою програмної інженерії стає сервісне програмування [1, 2]. Один з його актуальним викликів – автоматизована побудова й супровід сервіс-орієнтованих прикладних програмних систем (СоПС) на підтримку ділових процесів сучасних організацій. Саме вона є предметом досліджень колективу фахівців Інституту програмних систем (ІПС) НАН України під керівництвом директора ІПС, академіка НАН України, доктора фіз.-мат. наук П.І. Андона протягом останнього десятиріччя.

Для повно-аспектного опрацювання цього виклику прийнято низку установчих тез [3].

1. Узгоджене застосуванням інтелектуальних технологій і стандартів *семантичного* Вебу [1-4].

2. Конструювання СоПС згідно з де-факто стандартним *п'ятирівневим шаблоном* сервіс-орієнтованої архітектури [1–3, 5], показаним на рис. 1.

3. Конструювання третього (сервісного) рівня СоПС як *on-line композиції* виконуваних семантичних Веб-сервісів, відшукуваних в Інтернет.

4. Розмежоване дослідження підходів до компонування семантичних Веб-сервісів на *функціональному* та *процесному* рівнях [3, 6].

5. Застосування *мов* BPMN, BPEL та OWL-S, SWSL і SA-WSDL [3] для формального опису першого, другого й третього рівнів СоПС відповідно.

6. Збагачення ("семантизація") описів Веб-сервісів автоматично оброблюваними *семантичними анотаціями* [3, 6].

7. Застосування наукового доробку, отриманого в 1976 – 2012 рр. у руслі розвинутої концепції збіркового програмування у відділі програмної інженерії ІПС НАНУ під керівництвом доктора фіз.-мат. наук, професора К.М. Лавріщевої, до конструювання другого (компонентного) рівня СоПС та його подальший розвиток для сервісного рівня.

Наведені положення разом утворюють підґрунтя ефективного опрацювання загальних проблем інженерії складних розподілених програмних систем (ПС):

– інтероперабельності ($П_1$);

© П.І. Андон, О.О. Слабоспицька, 2017

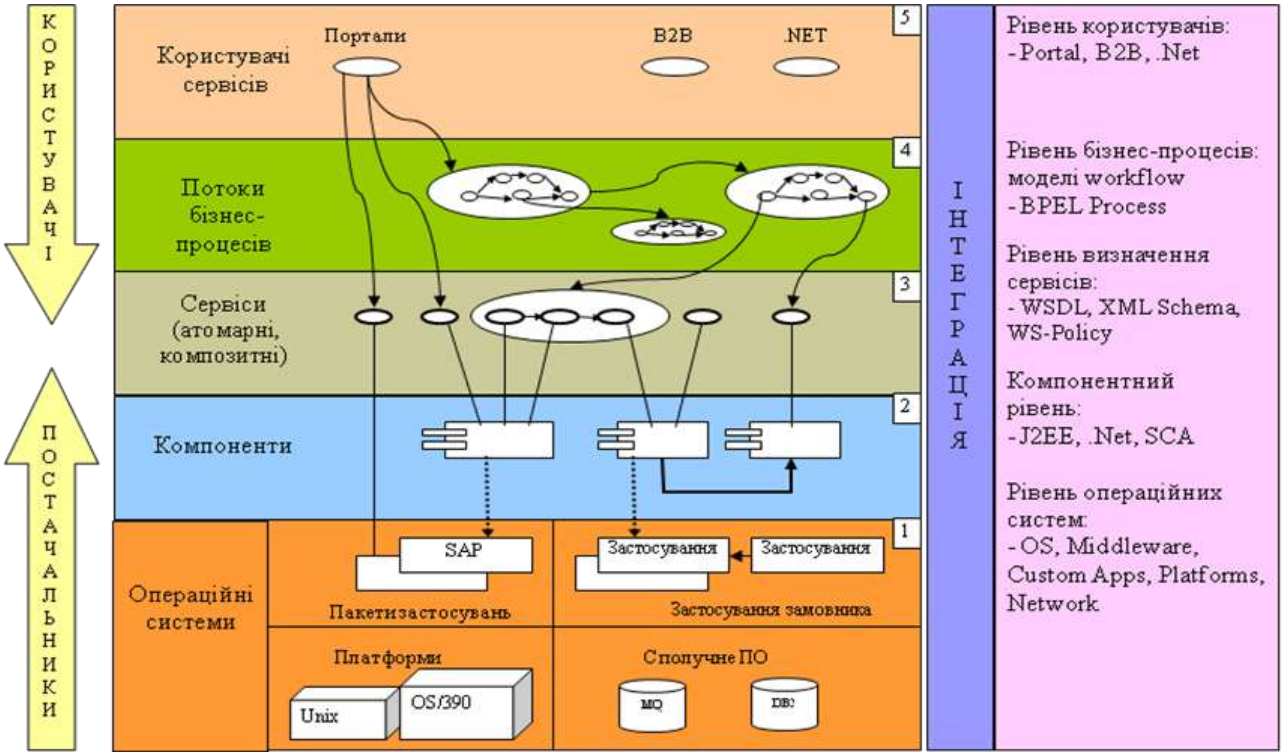


Рис. 1. П’ятирівневий шаблон сервіс-орієнтованої архітектури для СоПС

– повторної застосовності програмних ресурсів ($П_2$);

– змінюваності ПС за нових вимог та умов застосування ($П_3$);

– автоматизованого синтезу ПС з програмних ресурсів на підтримку заданих вимог ($П_4$);

– ресурсно-ефективного керування якістю ПС в її життєвому циклі ($П_5$).

У статті зроблено спробу простежити зв’язок парадигм компонентного та сервісного програмування під час опрацювання перерахованих проблем у побудові СоПС. Підсумовано актуальні результати та окреслено проблеми для подальшого опрацювання.

Формальний апарат збіркового програмування

Концепція збіркового програмування. Опрацювання зафіксованих проблем $П_1$–$П_5$ розробки СоПС в ІПС НАН України було розпочато на компонентному рівні СоПС (див. рис. 1). Розвинуто вітчизняну концепцію *збіркового* програмування складних (розподілених) ПС для певної предметної області (ПрО) [7–11]. Її сутність – втілення ідеї академіка В.М. Глушкова щодо автоматичного виробництва ПС з різнорідних готових ресурсів повторного використання (РПВ) на «збиральному конвеєрі» подібно до промислового конвеєрного виробництва. Таким чином, базовими поняттями збіркового програмування є:

1) РПВ – об’єкт зборки («деталь»);

2) метод зборки («конвеєр»);

3) інтерфейс між мовами програмування та між програмними РПВ («засіб з’єднання деталей»).

Метод збірки – спосіб організації необхідної взаємодії різнорідних програмних РПВ у структурованій ПС із взаємно однозначним перетворенням типів їх вхідних і вихідних даних. Міжмовний інтерфейс – сукупність формальних засобів цього перетворення для різномовних (і, можливо, різноплатформних) РПВ. Інтерфейс між програмними РПВ – сукупність формальних засобів організації обміну даними між ними.

З позицій збіркового програмування модульне, об’єктне, компонентне й сервісне програмування уніфіковано як парадигми *збіркового типу* [7–9]. У руслі перших трьох парадигм запропоновано формаліз-

ми для методу зборки відповідних РПВ та інтерфейсу, підсумовані в табл. 1 на підставі [9].

Як свідчить таблиця, кожна парадигма долає певні обмеження попередніх парадигм з опрацювання проблем $П_1$–$П_4$.

Крім того, подібність компонента повторного використання (КПВ) і Веб-сервісу, зокрема сервісу семантичного Вебу, зумовлюють перспективність вироблених в ІПС НАНУ рішень [1–3] з подальшого розвитку формального апарату компонентного програмування для автоматизованої підтримки життєвого циклу СоПС як (адаптивного) композитного сервісу семантичного Вебу. Ці рішення виділено в останньому стовпці таблиці й уточнено далі.

Таблиця

Доробок ІПС НАНУ в руслі парадигм програмування збіркового типу

Базове поняття	Парадигма програмування:			
	модульне (1976–1983)	об’єктне (з 1983)	компонентне (з 2000)	*сервісне* (з 2007)
1	2	3	4	5
Програмний РПВ – об’єкт зборки	Модуль – логічно завершений, незалежний і повторно застосовний фрагмент коду деякою мовою програмування для виконання певної функції [7–10]	Об’єкт – іменована частина дійсності з певним рівнем абстракції щодо цільової ПрО та цілком обумовленою поведінкою [7–9, 11], трактована як клас у сенсі Г. Буча [12]	КПВ – незалежна від МП програмна сутність із певними прикладними сервісами, доступними лише через інтерфейси, що визначають функції КПВ і спосіб звернення до них [7–11]	(Семантичний) Веб-сервіс – застосунок, незалежний від контексту й стану інших сервісів, ідентифікований URI, де інтерфейс і засоби зв’язування описані XML і семантично анотовані [3, 4, 6]
Метод зборки	Добір, розміщення в структурі цільової ПС і генерування посередників взаємодії модулів з перетворенням типів даних – на підставі паспортів модулів і ПС [7–10, 13, 14]	Реалізація композиції операцій взаємодії функціональних та інтерфейсних об’єктів згідно з об’єктним графом ПС [15]	Автоматичний добір мето-дів і їх віддалений виклик з необхідним перетворенням типів даних для КПВ, взаємодіючих згідно з компонентною моделлю ПС, за рахунок системних сервісів компонентного середовища [7–11, 16, 17]	Обмін повідомленя-ми між семантичними Веб-сервісами згідно з явним (через входи/виходи) чи неявним (через сервіс-оркестратор) графом композиції за HTTP, SOAP [3–6]
Міжмовний інтерфейс	Система процедур ізоморфного перетворення між заданими парами типів даних для фіксованих пар поширених МП ЄС ЕОМ [8, 13, 14]	Поповнена система процедур для складних типів об’єктно-орієнтованих МП (портфель, контейнер, протокол та ін.) [7–9, 15]	Системні сервіси компонентного середовища з реалізації поповненої системи процедур [7–9, 11, 17]	Засоби операційних систем і спеціалізованих двигунців для реалізації сервіс-орієнтованої архітектури в Семантичному Вебі [3]
Інтерфейс між РПВ	Система модулів-посередників, генерованих на підставі паспортів взаємодіючих модулів [13, 14]	Система інтерфейсних об’єктів [15]	Системні сервіси компонентного середовища з віддаленого виклику цільових методів [7–9, 11, 17]	Те ж саме

Закінчення таблиці

1	2	3	4	5
Подання РПВ	Паспорт спеціальною мовою [13, 14]	Клас у сенсі Г. Буча [12]	Кортеж описів вхідних і вихідних інтерфейсів (зазвичай мовою IDL), ідентифікаторів реалізацій певними МП і системних сервісів середовища [7–9, 16, 17]	Функціонального рівня – кортежі: входів/виходів (*IO*); *IO* з перед- і постумовами (*IOPE*), де елементам зіставлено концепти певної онтології [3]. Процесного рівня – BPEL- і WSDL-описи з семантичними анотаціями
Цільова ПС – результат зборки	Агрегат локальних модулів, структуру якого визначає паспорт ПС [13, 14]	Система функціональних та інтерфейсних об'єктів, взаємодіючих за об'єктним графом [9, 15]	Система (розподілених) КПВ, взаємодіючих згідно з компонентним графом за рахунок сервісів компонентного середовища [7–9, 11, 17]	СоПС – адаптивний динамічний композитний Веб-сервіс – система взаємодіючих сервісів, on-line відшукуваних і замінюваних для задоволення нових вимог і пристосування до нових (не)передбачених ситуацій [3, 18, 19]
Проблема	Частково – $П_1$, $П_2$, $П_4$	$П_1$, частково $П_2$, $П_4$	$П_1$, $П_2$, $П_4$, частково $П_3$	$П_2$–$П_4$, частково $П_1$, $П_5$
Недолік	Реактивне повторне використання, локальність, одномовність модулів, спеціальні засоби взаємодії	Потреба ручного зв'язування з вимогами, одномовність об'єктів, спеціальні засоби взаємодії	Спеціальні засоби взаємодії, надавані системними сервісами компонентних середовищ, обмеженість змін під час виконання	Труднощі гарантування якості сервісу, опрацювання Big Data, керування мережевими ресурсами

Поряд з формалізмами окремих парадигм, для застосування методів зборки запропоновано універсальні формальні засоби перетворення типів вхідних/вихідних даних взаємодіючих різномовних (і, можливо, різноплатформних) РПВ [7–11]:

– подання типу даних певної мови програмування (МП) алгебраїчною системою, де носій – множина значень типу, а сигнатура – перелік операцій на носії;

– формалізацію перетворення типів даних як ізоморфізму відповідних їм алгебраїчних систем;

– алгоритми взаємно однозначного перетворення між фундаментальними типами даних МП (простими й складними) та загальними типами даних стандарту ISO/IEC 11404:2007 (примітивними, агрегатними і генерованими).

Запропоновані засоби налаштовано для модульного, об'єктного й компонентного програмування згідно з особливостями відповідних РПВ (див. таблицю).

Модульне програмування. У цій парадигмі створено систему автоматизації виробництва програм (АПРОП) [13, 14] –

одну з перших «фабрик програм» академіка В.М. Глушкова. В ній реалізовано:

– технологію побудови банка модулів для автоматизації розв'язання задач певної ПрО та опису модулів спеціальними паспортами;

– бібліотеку з 64 інтерфейсних функцій, кожна з яких реалізує припустиме перетворення між типами даних поширених МП ОС ЕС (Фортран, Алгол, Кобол, ПЛ/1, Асемблер);

– спеціальний модуль – *монітор*, що реалізує метод зборки та міжмодульний і міжмовний інтерфейси (за допомогою генерування інтерфейсних модулів-посередників з операторами виклику необхідних інтерфейсних функцій для обміну даними між взаємодіючими модулями).

Об'єктне програмування. Цю парадигму розвинуто формалізмами функціонального й інтерфейсного об'єктів (ще трактованих як класи в сенсі Г. Буча [12]). Вони є «наступниками» цільового модуля та модуля-посередника в модульному програмуванні. *Функціональний* об'єкт, локальний або віддалений в гетерогенному середовищі, – формальне подання очікуваних функцій (розподіленої) ПС (РПС), які забезпечать розв'язання задач певної ПрО. *Інтерфейсний об'єкт* – подання операцій виклику методів функціональних об'єктів і необхідного перетворення типів їх вхідних/вихідних даних, що опосередковує взаємодію функціональних об'єктів. Побудовано алгебру операцій взаємодії функціонального й інтерфейсного об'єктів між собою та з об'єктним середовищем [7, 15]. Надані формалізми підтримують підвищення рівня адекватності проектування розподілених ПС та повторного використання об'єктів за рахунок оптимізації об'єктного графу для ПрО.

Компонентне програмування. Розглянуті формалізми об'єктного програмування розвинуто для опрацювання їх обмежень – зорієнтованості на конкретну розподілену ПС (РПС) і певну МП, неможливість змінення без доступу до коду.

Центральним складником розробленого формального апарату є *об'єктно-компонентний метод* проектування ПС (ОКМ) К. Лавріщевої, В. Грищенка [8–11, 16, 17]. Сутність ОКМ – подання цільового домену (тобто всіх ПС на підтримку потреб суб'єктів цільової ПрО) трійкою взаємопов'язаних моделей – об'єктною, інтерфейсною, компонентною. Ці моделі являють собою графи, вершинами яких є запропоновані моделі [9,11,16] відповідно об'єктів ПрО, їх інтерфейсів і КПВ реалізації інтерфейсів, а дугами – класифікаційні й предметні відношення між вершинами. Отже, модель окремої (розподіленої) ПС – трійка відповідних підграфів.

Для об'єкта запропоновано [16] первинне подання *трикутником* Г. Фреге. Він пов'язує:

– знак – ідентифікатор, що позначає певну сутність об'єктивної дійсності;

– денотат – позначену сутність;

– концепт – сукупність значущих для проектувальника домену властивостей цієї сутності, тобто предикатів, які можна отримати з множини унарних предикатів "мати властивість", істинних для неї, за допомогою логічних зв'язок.

Передбачено уточнення таких "трикутників" до класу в сенсі Г. Буча.

Інтерфейс подано унікальним ім'ям і переліком сигнатур методів із специфікаціями їх вхідних/вихідних даних. Моделлю КПВ є кортеж, утворений унікальним ім'ям, множинами вхідних і вихідних інтерфейсів та реалізацій їх методів певними МП, інтерфейсом керування екземплярами КПВ і множиною системних сервісів.

ОКМ охоплює два кроки: об'єктний аналіз домену та проектування КПВ.

Мета об'єктного аналізу – подати домен графом функціональних та інтерфейсних об'єктів – класів у сенсі Г. Буча, методи яких реалізують усі функції можливих ПС, запитані суб'єктами ПрО, та їх взаємодії з необхідним перетворенням даних. Відповідно, дуги подають відношення взаємодії між об'єктами.

Для цього послідовно формують підграфи трьох проміжних рівнів [9, 11, 16]:

– узагальнюючого, де вершинами є об'єкти – трикутники Фреге з невизначеними концептами, значущі для суб'єкта аналізу, а множина дуг порожня;

– структурно-впорядковуючого, де вершини тієї ж природи, а дуги подають відношення Is_a й Part_of між ними на підставі їх денотатів у теоретико-множинному сенсі;

– характеристичного, де формують концепти для вершин, уточнюють відношення Is_a й Part_of та фіксують предметні відношення між ними згідно зі складом концептів.

Нарешті, на четвертому поведінковому рівні на підставі сформованих концептів визначають методи об'єктів, формують їх інтерфейси й зіставляють кожному трикутнику Фреге один чи декілька класів. Складають послідовності станів отриманих класів і діаграми переходів станів. Залежність від часу враховано долученням до формованого підграфа спеціального об'єкта *Timer* для надсилання повідомлень про поточний час за певним регламентом. Решта об'єктів мають відповідний метод, що аналізує отримане значення та, за потреби, ініціює перехід до іншого стану.

На підтримку адекватного формування зазначених рівнів запропоновано двоосновну *алгебраїчну систему об'єктного аналізу* [9, 16]. Вона містить:

– множини подань об'єктів (трикутником Фреге і класом);

– операції декомпозиційної зміни денотату об'єкта для формування нових (не)однорідних об'єктів;

– операції композиційної зміни денотатів (не)однорідних об'єктів для формування нового об'єкта;

– операції розширення/звуження концептів наявних об'єктів;

– унарні відношення "мати властивість", відповідні значущим властивостям об'єктів ПрО;

– відношення Is_a, Part_of й істотні класифікаційні відношення об'єктів ПрО.

У свою чергу, проектування КПВ передбачає ізоморфне відображення побудованого графу класів у граф моделей їх інтерфейсів та побудову сукупності моделей компонентів, які разом ресурсно ефективно реалізують методи інтерфейсів.

Для розширення сфери застосування ОКМ і підвищення його ефективності запропоновано решту формалізмів [9, 11, 16]:

– моделі компонентної програми разом з умовами її цілісності та компонентного середовища;

– відношення тотожності, еквівалентності, подібності, успадкування, контракту компонентів і зв'язування їх екземплярів;

– двоосновну *зовнішню компонентну алгебру*, що поєднує операції розгортання компонента в компонентному середовищі, (комутативне й асоціативне) об'єднання середовищ і вилучення компонента з середовища;

– *внутрішню алгебру рефакторингу* компонента з операціями долучення нової реалізації для існуючого інтерфейсу й разом з новим інтерфейсом, заміщення існуючої реалізації новою без зміни вхідного інтерфейсу, долучення нового вхідного інтерфейсу для існуючої реалізації;

– *алгебру реінжинірингу* компонента, де операції рефакторингу доповнено вилученням інтерфейсу, зміненням його сигнатури, вилученням реалізації;

– багатоосновну *алгебру* операцій взаємного *ізоморфного перетворення* фундаментальних типів даних поширених МП (простих, структурованих і складних *неструктурованих*) між собою та між загальними типами ISO/IEC 11404:2007 під час взаємодії різномовних і/або різноплатформних КПВ.

Зазначені формалізми інтегровано в інформаційній технології *автоматизованої зборки* ПС для певної ПрО з попередньо відшуканих/розроблених КПВ. Метод зборки реалізовано ресурсно ефективною композицією операцій компонентних алгебр та алгебри реінжинірингу за компонентною моделлю ПС.

Таким чином, запропонований апарат компонентного програмування вперше

уможливлює *проактивне* повторне використання КПВ за рахунок формального зв'язку очікуваних функцій РПС з інтерфейсами КПВ їх реалізації, запровадженого в ОКМ, і змінення компонентної РПС без доступу до коду операціями запропонованих алгебр [9, 11, 16]. Однак потребу й аргументи цих операцій усе ще має визначати користувач/супроводжувач РПС у ручному режимі на підставі внутрішньої структури КПВ.

Значна трудомісткість і тривалість цього все ще "ручного" змінення та висока ймовірність незадовільної якості зміненої РПС через помилки його виконавця зумовили зосередження подальших досліджень на формальних засобах збіркового програмування автоматизовано змінюваних та адаптовних РПС.

Забезпечення змінюваності складних РПС

Формальні засоби забезпечення змінюваності. Створені засоби є результатом узгодженого розвитку [20–24] двох загальновизнаних підходів до індустріально-го виробництва змінюваних ПС: генерувального програмування (K. Czarnecki) та побудови ліній програмних продуктів (K. Pohl). Їх об'єднує перехід від розгляду окремої ПС до Сімейства ПС (СПС). СПС – це поповнюваний набір ПС, елементи якого мають керовану множину спільних *властивостей* на підтримку потреб споживачів у ПрО й розробляються попередньо визначеним способом із спільної множини РПВ[1]. Властивість – вимога, функція чи нефункціональна характеристика, запитана споживачами.

Незалежно від підходу, визначальними особливостями СПС є [21, 22]:

– простори *Проблеми* й *Рішень*, утворені передбаченими для ПрО обов'язковими та змінними властивостями РПС і, відповідно, РПВ з правилами їх компонування в РПС для реалізації властивостей;

– формалізм *моделі властивостей* (МВ) – їх дерева з відношеннями обумовленості, зіставлення, виключення й варіантного підпорядкуванням [21], що є де-факто стандартом опису простору проблеми;

– процеси конструювання *домену* (де розробляють РПВ) і *застосунків* (де РПВ компонують у ПС із спільними й передбаченими змінними властивостями);

– процес керування варіабельністю – здатністю ПС до ефективного розвитку, зміни, налаштування або конфігурування для використання в певному контексті, проявленої *точками варіантності* (проявами артефактів, реалізованих кількома способами) та їх варіантами [23].

Зафіксовано критичне обмеження обох підходів – відсутність формалізму зв'язування властивостей ПС з ресурсами їх реалізації. Саме воно вимагає "ручного" компонування РПВ або визначення генеруючої моделі [20].

Для подолання цього обмеження:

– розроблено моделі *варіабельності в структурі й артефактах* СПС [21, 23], що поширюють МВ на основні артефакти процесу розробки (вимоги, архітектуру, програмні РПВ тощо) за допомогою формалізованих *зв'язків трасовності*, та модель *оцінювання рівня варіабельності* [23];

– за допомогою ОКМ першу модель перетворено в *об'єктно-компонентну модель варіабельності* змінюваної компонентної РПС, розглядуваної як сімейство її передбачених варіантів [21] і сімейства таких РПС [22], а другу – в подібну модель передбаченого *варіанта* компонентної РПС або члена їх сімейства. В цих моделях канонічні для ОКМ графи об'єктів, інтерфейсів і КПВ доповнено формалізованими точками варіабельності й їх варіантами, що описують припустимі зміни функцій РПС. Для побудови такого графу "варіабельних об'єктів" розвинуто багатотуровий *Процес конструктивного аналізу* ПрО [22] з використанням технік Інженерії співпраці для експертного зіставлення властивостей ПрО потенційним РПС;

– запропоновано рамкову модель *сімейства змінюваних* РПС, зокрема компонентних [21], складником якої є зазна-

[1] http://www.sei.cmu.edu/productlines/

чена модель *варіабельності в структурі* зокрема, її об'єктно-компонентне уточнення), яка уможливлює автоматизоване змінне компонування РПВ згідно з визначеними в ній зв'язками трасовності;

– третю модель доповнено рамковим переліком проявів незадовільної варіабельності сімейства змінних компонентних РПС і перетворено в *діагностичну* модель оцінювання рівня варіабельності компонентних РПС та їх СПС [22].

На підставі наданої моделі змінної компонентної РПС формалізовано операції її *первинного* компонування (з КПВ і/або інших РПС) і (*не*)*передбаченого змінення* для адаптації до нових вимог й умов вико-нання. Показано, що ці змінення (шляхом спеціального вилучення, долучення й заміни варіантів у точках варіабельності об'єктів і КПВ) утворюють двоосновні алгебри (не)передбачених варіантів РПС [22]. Надані алгебри являють собою розвиток алгебр рефакторингу й реінжинірингу в ОКМ для складених змінюваних КПВ.

Побудовано рамкову технологічну модель процесу *автоматизованого виробництва* СПС [24]. Його подано композицією операцій запроваджених функцій *обґрунтованого керування* варіабельністю [23–26] у середовищі трьох вищезазначених рамкових моделей згідно з генерувальною моделлю. Вона визначена як відображення між просторами проблеми й рішень відповідно до зв'язків трасовності у моделі варіабельності в структурі СПС.

Перелік функцій узагальнює дії у відомому циклі PDCA Е. Дьомінга в індустріальних технологіях розроблення СПС. Він охоплює: технологічну підготовку й ініціалізацію модельного середовища, планування варіабельності в структурі й артефактах СПС, її реалізацію, відстеження задовільності й актуалізацію СПС для опрацювання незадовільності.

Технологія конфігураційного збирання змінюваних РПС. На підтримку конструювання РПС з РПВ (зокрема, з КПВ) запропоновано [21, 25, 26] інтегрувати процес керування конфігурацією РПВ до спеціального процесу *проактивного обґрунтованого керування варіабельністю* РПС. Він є технологічним уточненням розглянутого процесу виробництва СПС [24] за рахунок надання функціям керування варіабельністю спеціального змісту:

– планування складу РПС і РПВ (КПВ) з їх варіантами для цільової ПрО;

– добір, налаштування або безпосередня розробка термінальних РПВ (КПВ) та їх автоматизоване компонування ("збирання") у цільові РПС за моделлю варіабельності в структурі чи її ОКМ-уточненням;

– відстеження відповідності складу РПС і РПВ (КПВ) потребам споживачів з діагностуванням невідповідностей і вироблення коригуючих дій щодо відповідних артефактів;

– актуалізація моделі варіабельності в структурі (її ОКМ-уточнення) і/або поточного складу РПС і РПВ (КПВ) для опрацювання незадовільності.

На підтримку виконання наведених функцій, вищезазначене модельне середовище доповнено репозиторіями [25, 26]: РПВ (КПВ); профілів варіабельності; постановок і апробованих методів для оптимізаційних задач ресурсно ефективного планування, експертно-аналітичного оцінювання [27] та ідентифікації структури багатовимірних даних; показових метрик варіабельності.

Для другої функції – реалізації варіабельності запропоновано архітектурні рішення спеціалізованого інструментального засобу, Конфігуратора РПС [25, 26]. Він надає зацікавленим особам інтерфейс автоматизованого створення РПС із заданими функціями з наперед розроблених КПВ або змінення існуючої РПС.

Технологічна підтримка формального апарата

Інженерія якості ПС. Для цільового опрацювання проблеми $П_5$ незалежно від парадигм програмування запропоновано *парадигму якості* ПС і відповідну їй концепцію *Інженерії якості* [28]. Їх визначальна особливість – акцент на *випере-*

джуючому керуванні якістю в життєвому циклі (ЖЦ) ПС, *узгодженому* на всіх рівнях керівних впливів (від організації у цілому до функціональних проектних груп) і для всіх об'єктів ЖЦ, якість яких можна визначити, виміряти й підвищувати (від процесів ЖЦ до окремих артефактів програмних проектів).

Інженерію якості визначено [28] як інтегрований процес забезпечення для всіх релевантних об'єктів ЖЦ на кожній його стадії певних властивостей, що характеризують їх якість, які можна прогнозувати, вимірювати і вдосконалювати за допомогою системи спеціальних процесів, методів і технологічних засобів.

На підтримку інженерії якості:

а) сформовано відповідне *Ядро знань* [28], що систематизує кращі практики керування якістю в ЖЦ ПС та постановки й методи розв'язання задач на їх підтримку відповідно до стадій ЖЦ ПС на підставі відомих ядер знань з програмної інженерії (SWEBOK) та проектного менеджменту (PMBOK);

б) запропоновано оригінальний *каркас інженерії надійності* ПС [29–32]. Він містить постановки й методи розв'язання на стадіях ЖЦ ПС її основних задач – від розподілу вимог до надійності за логічними модулями ПС до оптимізації витрат на усунення причин виявлених дефектів, із застосуванням апарату мереж Байєса й методу аналізу ієрархій Т. Сааті;

в) розроблено технологічну модель процесу *ресурсно ефективного тестування* різнорідних ПС за *умов обмежених ресурсів* [33], яка інтегрує цілі, задачі й дії з тестування, розподілені в ДСТУ ISO/IEC 12207 серед процесів ЖЦ. Поставлено й розв'язано, для основних моделей зростання надійності, задачу визначення оптимального моменту випуску ПС, для якого ризик її подальшої відмови мінімізується, але витрати на тестування ще не перевищують можливого збитку в разі відмови;

г) розроблено математичну модель нового процесу *обґрунтованого експертного оцінювання* об'єктів ЖЦ ПС, запропонованого до інтеграції в ЖЦ, та систему методів узгодженої підтримки його операцій [27]. Модель поєднує подання його інформаційного й технологічного середовища та підпроцеси розв'язання задач з операціями підготовки, проведення й інтеграції результатів взаємопов'язаних експертиз оцінюваних об'єктів ЖЦ ПС. Запровадження процесу уможливлює створення баз досвіду розроблення ПС на необхідних організаційних рівнях і створює передумови вдосконалення процесів ЖЦ ПС й підвищення ефективності дій їх учасників;

д) надано рамкову технологію *інтелектуального керування* значущими показниками об'єктів ЖЦ (ризиком, витратами, зрілістю процесів тощо), технологічне й математичне ядро якої складає розроблений формальний апарат експертного оцінювання. Уточнено постановки задач і надано методи їх розв'язання для ризику невиконання програмного проекту [34];

е) побудовано онтологію ПрО тестування засобами редактора онтологій Protégé [20].

Розвинуто засади інженерії якості СПС [20, 35]:

– запропоновано рамкову модель його якості разом з варіабельністю як ключовим чинником впливу на решту характеристик;

– надано систему показників якості архітектури СПС та уточнено для них процес експертного оцінювання із застосуванням взаємопов'язаних оцінних моделей – аргументованого дерева цінності, мережі Байєса, ієрархії Т. Сааті – у відповідних ситуаціях оцінювання;

– розроблено рамкові процедури підвищення гнучкості й ефективності *співпраці* учасників *процесу аналізу ПрО* для формування простору проблеми СПС за допомогою гнучких методологій та уніфікованих шаблонів елементарних переговорних дій (так званих *сінклетів*) з арсеналу Інженерії співпраці [22].

Стандартизація в програмній інженерії. Для надання нормативно-методичної підтримки вирішення вищеза-

значених проблем $П_1$–$П_5$ побудови ПС фахівцями ІПС:

– розроблено понад 30 спеціалізованих ДСТУ серій «Інформаційні технології» та «Програмна інженерія», гармонізованих з міжнародними (зокрема, груп 15504, 14143, 9126, 250XX). Вони технологічно й методично уточнюють описи відповідних процесів у еталонній моделі ЖЦ за ДСТУ ISO/IEC 12207:2014;

– створено понад 20 методик з обґрунтованого спадкоємного оцінювання значущих показників об'єктів ЖЦ ПС і керування ними (зокрема, функціонального розміру ПС, ризику невиконання програмного проекту, технологічної зрілості процесів ЖЦ, витрат на розроблення ПС);

– визначено склад інструментів підтримки виконання, оцінювання й удосконалення процесів і розроблено макети ПС на підтримку окремих методик з оцінювання трудомісткості й ризику. Удосконалені версії цих ПС і досі мають попит.

Для розбудови та вдосконалення національної системи стандартизації в галузі програмування, інформаційних технологій та програмної інженерії в ІПС запропоновано інструментально- та організаційно-орієнтовані моделі стандартизації [36], зокрема, принципово нові, повторну, часткову, зорієнтовану на генеровану програмну продукцію, інтегровану. Запропоновано узагальнену концептуальну модель стандартизації документування комп'ютерних програм, яка містить вимогу використання генерованого документування. На її основі розроблено склад першої черги «Єдиної системи стандартів документування комп'ютерних програм», наданої Держспоживстандарту як пропозиції до Державного плану стандартизації.

Апробація теоретичного доробку

Практичну значущість розроблених формальних засобів засвідчено їх успішною різноаспектною апробацією.

Систему АПРОП успішно впроваджено в 52 організаціях СРСР. Її також застосовано для виробництва вбудованих бортових програм як базовий складник інструментально-технологічних комплексів "Прометей" і "РУЗА" [13, 14], що поставлялися в галузеві організації Міністерства радіопромисловості СРСР до 1991 р.

Розроблені формальні й технологічні засоби компонентного програмування РПС, зокрема макетний зразок Конфігуратора і технологічну схему збирання КПВ за його допомогою, реалізовано в інструментально-технологічному комплексі виробництва програм ІПС НАНУ (ІТК) (http://sestudy.edu-ua.net) [37] та експериментальній фабриці програм КНУ ім. Т. Шевченка (programsfactory.univ.kiev.ua).

Як показано на рис. 2, реалізація комплексної технології збирання змінюваних РПС з КПВ в ІТК охоплює вкладені технології, подані у форматі технологічних ліній [20]:

– проектування РПС з урахуванням їх ЖЦ за ДСТУ ISO/IEC 12207:2014;

– збереження результатів проектування в репозиторії КПВ;

– онтологічного опису доменів;

– специфікації різнорідних програмних РПВ відповідними МП та їх накопичення в репозиторії;

– вибору КПВ у репозиторії;

– конфігураційного збирання різнорідних КПВ у (змінювану) ПС з перетворенням даних для них;

– тестування КПВ і РПС, збирання даних для контролювання їх якості.

Для реалізації процедур перетворення даних в ІТК передбачено бібліотеку функцій перетворення між загальними (примітивними, агрегатними й генерованими) та фундаментальними (простими, структурними і складними) типами даних для поширених МП, необхідних у гетерогенному середовищі взаємодії різномовних КПВ і РПС.

ІТК і студентську фабрику успішно застосовують на факультеті комп'ютерних наук та кібернетики КНУ ім. Т. Шевченка у навчанні студентів технологіям програмної інженерії [20].

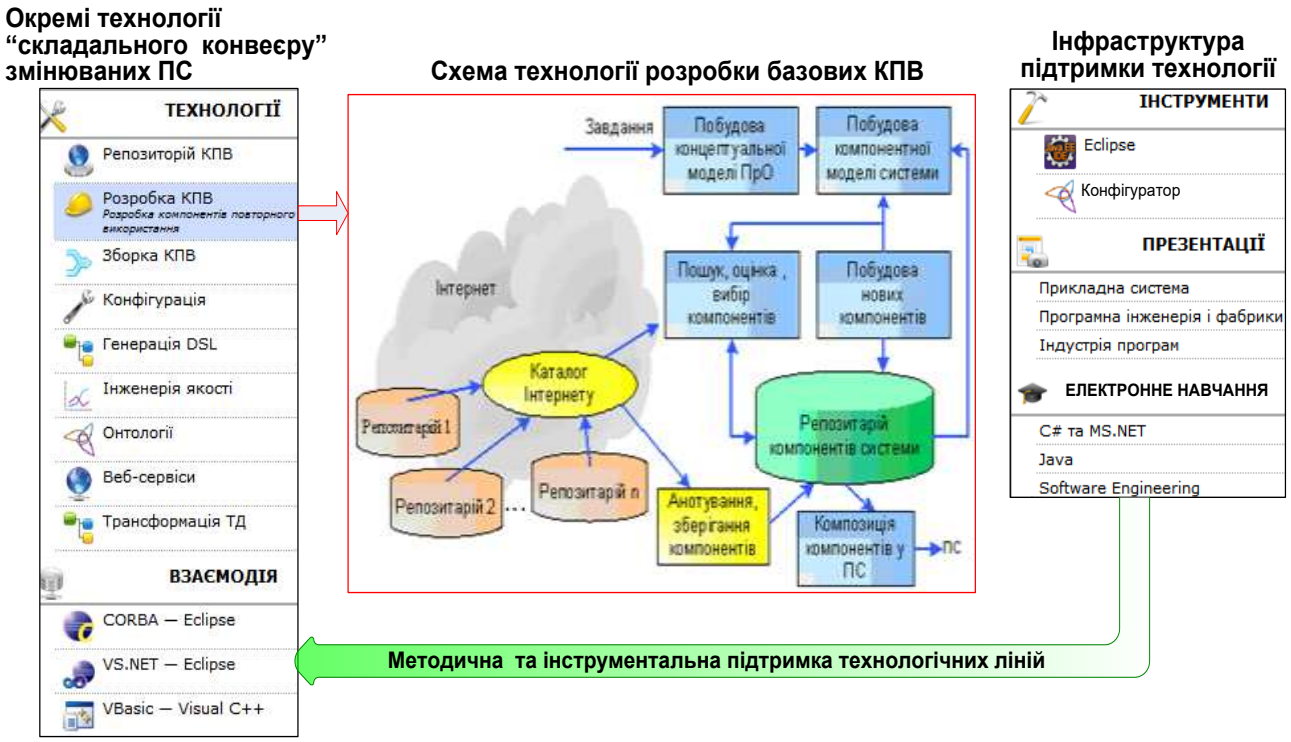


Рис. 2. Структура технологій ІТК ІПС НАН України

Формальний апарат підтримки життєвого циклу СоПС

Уточнення засад досліджень. Технічні рішення щодо семантичних Веб-сервісів у ролі РПВ, окреслені в таблиці, зумовлюють їх істотні переваги в опрацюванні проблем $П_1$–$П_5$ для СоПС: загальнодоступність, слабке зв'язування, можливість автоматично взаємодіяти незалежно від платформи і, завдяки протоколу HTTP, через міжмережний екран. Однак необхідною передумовою досягнення переваг є збагачення описів Веб-сервісів семантичними анотаціями функціонального й процесного рівня із застосуванням стандартів саме семантичного Вебу [4] згідно з положеннями $Т_1$, $Т_4$. Ці анотації уможливлюють автоматичне зіставлення як входів і виходів, так і додаткових складників опису семантичних Веб-сервісів, на якому ґрунтуються розвинуті алгоритми їх автоматичного компонування, надані далі. Таким чином, *програмування в семантичному Вебі* можна визначити як традиційне сервісне програмування, але з анотаціями.

Семантичний веб – це загальнодоступна глобальна семантична мережа, формована на основі мережі Інтернет за допомогою стандартизованого подання інформації, придатного для машинного оброблення. Автоматичний аналіз даних, виведення висновків, перетворення даних і висновків у практично корисні артефакти в Семантичному вебі можливі завдяки двом його особливостям: наявності уніфікованого ідентифікатору ресурсів (URI) [3, 4] та використанню онтологій, описаних стандартизованими мовами.

URI – це Інтернет-адреса, яку застосовують для однозначного машинно-сприйманого посилання на довільний об'єкт (веб-сторінку, файл, скриньку електронної пошти, художній твір, абстрактне поняття тощо). Унікальнсть URI уможливлює збирання інформації про деякий об'єкт з рідних джерел.

Основними складниками "стеку" стандартів для мов опису онтологій семантичного Вебу є *XML*, *XML Schema*, *RDF*, *RDF Schema*, *OWL* [3].

XML надає синтаксис для визначення структури документа, що підлягає машинній обробці, та не має семантичного навантаження. *XML Schema* визначає об-

меження на структуру XML-документа, дотримання яких можна перевірити за допомогою стандартного синтаксичного аналізатору мови XML.

RDF надає простий спосіб опису екземплярів даних у форматі *суб'єкт-відношення-об'єкт*, причому в цій трійці застосовують тільки ідентифікатори ресурсів (крім об'єкта, який може бути літералом). Стандартизовано відображення цих трійок в XML-документи визначеної структури. *RDF Schema* описує набір атрибутів (відношень) для визначення нових типів даних, описаних *RDF*. Мовою підтримано також відношення спадкування типів таких даних.

OWL надає додаткові можливості, розширює можливості опису нових типів (зокрема, долученням перерахувань), а також дозволяє описувати нові типи даних RDF Schema в термінах вже існуючих (наприклад, визначати тип, який є *перетином* або *об'єднанням* двох існуючих). Мова ґрунтується на дескриптивній логіці й поєднує три вкладені підмножини: *OWL Lite*, *OWL DL* і *OWL Full*.

SPARQL дозволяє запитувати дані у форматі RDF і формувати протоколи для передавання запитів і отримання відповідей на них.

Таким чином, на підставі окресленого уточнення установчих тез $Т_1$–$Т_6$ поставлено спеціальні питання опрацювання проблем $П_1$–$П_5$ для СоПС і надано поточні конструктивні відповіді на них [3].

У відповідь на питання *специфікації* (як визначити однозначно синтаксис і семантику Веб-сервісу?) для подальшого аналізу відібрано множину перспективних засобів його опису [6]: *OWL-S*[2], каркас *SWSF*[3] з онтологією *SWSO* й мовою *SWSL*, онтологію *WSMO*[4] з мовою *WSML*, мови *RDF* і *RDF-S*, *WSDL-S i SA-WSDL*[5]. Відповіддю на питання *публікування* (як зробити доступною інформацію про веб-сервіси?) є застосування реєстрів сервісів із специфікацією UDDI та її семантичних розширень. Для відповідей на питання *виявлення* (як знайти прийнятний(і) сервіс(и) для певного завдання?) та *вибору* (як знайти серед них найкращий?) відібрано базові критерії семантичного метчмейкінгу [18, 38] – збіг, включення, вміщення, несумісність та показники якості Веб-сервісів (QoS) й алгоритми хешування [18].

Питання компонування семантичних Веб-сервісів (як зібрати кілька атомарних і композитних сервісів для конкретної роботи?) передбачено опрацювати насамперед за допомогою графо-орієнтованих підходів функціонального рівня [38], які ґрунтуються на поданні Веб-сервісів у форматі (*IO*) й (*IOPE*). Далі отриманий доробок буде розвинуто для процесного рівня за рахунок універсальних технік планування через перевірку моделей [19]. Ці ж техніки й відповідні їм інструментальні засоби обрано як підґрунтя вирішення питання верифікації (як перевірити атомарні й композитні веб-сервіси?).

Для відповіді на чергове питання *інтероперабельності* (як долати різнорідність даних, що ними обмінюються веб-сервіси) обрано перспективні підходи Extract, Transform, Load[6] і Linked Data[7].

Для вирішення питань *взаємодії* запитувачів і постачальників сервісів (які дані й посередники протоколу потрібні для них?) і *посередництва* (як забезпечити прямий доступ запитувачів до опублікованих сервісів) вибрано специфікацію UDDI з її семантичними розширеннями та подання функцій композитних сервісів моделлю властивостей для їх семантичного анотування й публікування.

Відповідь на питання *моніторингу/керування* (як керувати мережними ресурсами, гарантуючи QoS?) надають сучасні підходи дискретної оптимізації на графах. Нарешті, відповіддю на останнє питання *виконання* (як надійно виконати семантичний Веб-сервіс) є розвиток окресленого вище доробку Інженерії

[2] http://www.w3.org/Submission/OWL-S/
[3] http://www.w3.org/Submission/SWSF-SWSO/
[4] http://www.w3.org/Submission/WSMO-primer/
[5] https://www.w3.org/TR/2007/NOTE-sawsdl-guide-20070828/
[6] http://www.dataintegration.info/etl
[7] http://linkeddata.org/

якості та забезпечення його адаптивності до відмов.

Компонування семантичних Веб-сервісів. На підтримку життєвого циклу (ЖЦ) СоПС розроблено його високорівневу схему як композиції базових задач, розв'язуваних у семантичному Вебі із застосуванням вище зафіксованих відповідей, показану на рис. 3. Згідно з рисунком, центральну роль у ЖЦ СоПС відіграють взаємопов'язані задачі on-line виявлення й компонування семантичних

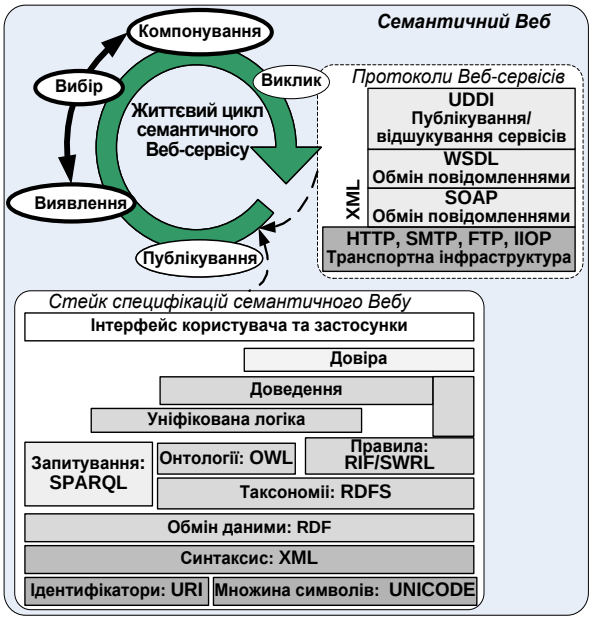


Рис. 3. Схема ЖЦ СоПС

Веб-сервісів. Наразі їх досліджено в найпростішій постановці функціонального рівня з використанням каркасу P. Rodriguez-Mier і C. Pedrinaci [2, 3, 38, 39]. Його вибір зумовлено перевагами порівняно з аналогами:

– мінімальною втратою продуктивності між виявленням і композицією;

– відсутністю припущень про локальну доступність сервісів і завантаження їх реєстрів у пам'ять двигунця пошуку;

– наявністю оптимального алгоритму для вилучення з графа найкращої композиції, що мінімізує кількість сервісів, та алгоритмів оптимізації графа для підвищення масштабовності формованого композитного Веб-сервісу [39].

Досліджувана постановка має вигляд:

для запиту $r=\left(In_r, Out_r\right)$, складеного множинами доступних вхідних і вихідних концептів деякої онтології O, і множини $W=\{w_j=(In_j, Out_j), j\geq 1\}$ семантичних Веб-сервісів з реєстрів Інтернет, теж поданих парами вхідних і вихідних концептів O, знайти динамічний композитний Веб-сервіс

$$w_c=((In_{wc}, Out_{wc}),\ P=(S,\leq)),\ S\subseteq W, \quad (1)$$

такий, що

$$In_r \otimes In_{wc}=In_{wc};\ Out_{wc}\otimes Out_r=Out_r, \quad (2)$$

де $\otimes$ – оператор метчмейкінгу [38], що зіставляє парі множин концептів підмножину першої з них, елементи якої пов'язані з кожним концептом другої відношенням збігу або включення;

$\leq$ – відношення часткового порядку на S, що неявно подає послідовність виконання компонентних сервісів.

Таким чином, формований сервіс w_c має викликатися за запитаними входами In_r й надавати запитані виходи Out_r. Оригінальний алгоритм *компонування від входів* (P. Rodriguez-Mier) [38] забезпечує подання w_c графом композиції – орієнтованим ациклічним графом, де вершинами є компонентні сервіси $w_j\in W$ і вхідні й вихідні концепти для них, а ребра поєднують: вхідні, що з'єднують вхідні концепти з їх сервісами; вихідні, що зіставляють сервісам вихідні концепти, та проміжні, що подають відношення збігу або включення концептів. Цей граф є *ярусно-паралельним* [3], тобто його розбито на шари (яруси), кожний з яких містить усі ті сервіси, входи яких відповідають, у сенсі оператора $\otimes$, виходам сервісів попереднього шару. Граф доповнено нульовим і додатковим ярусами, які містять технічні сервіси ($\varnothing$, In_r) і (In_r, $\varnothing$).

Наданий алгоритм передбачає розмежоване виявлення сервісів, семантично відповідних запитаним множинам входів In_r і виходів Out_r, із застосуванням лише деяких/довільних їх елементів. Тому граф композиції первинно є надлишковим і містить усі сервіси з W, які можна безпосередньо чи опосередковано викликати за запитаними входами In_r. Однак алгоритм аж ніяк не гарантує, що всі гілки графа досягнуть запитаних виходів. Тому запропонований каркас передбачає подальшу оптимізацію первинного графа композиції для перетворення її на припустиму або оптимальну з найменшою кількістю складників для заданого рівня QoS, які й становлять інтерес для конструювання СоПС.

Композицію, задану сервісом w_c (1), (2) для запиту $r=\{In_r, Out_r\}$, звуть *припустимою*, коли кожний її сервіс можна викликати, тобто у графі композиції немає гілок, не пов'язаних з деяким вхідним концептом – елементом In_r.

Ефективний за "розумної" розмірності множини In_r, алгоритм компонування "від входів" може стати незручним, коли вона дуже мала або дуже велика. Тому запропоновано його розвиток – побудову w_c (1), (2) "від виходів" [3]. Вона особливо доцільна за неприйнятної розмірності множини In_r і прийнятної – Out_r. Розроблений алгоритм являє собою "обернення" оригінального алгоритму для припустимої композиції [38]. Результат його застосування показано на рис. 4.

Алгоритм вибирає з W усі сервіси, що надають виходи, відповідні вхідним концептам кожного ярусу. Далі для кожного вибраного сервісу зіставляються наявні концепти та його незіставлені виходи. Всі виходи, які зіставлено, вилучають з неузгодженого набору виходів для поточного сервісу. Якщо немає незіставлених виходів, то сервіс є викличним і, отже, прийнятним у поточному ярусі. Наприклад, для композиції на рис. 4 першими прийнятними є сервіси в ярусі L_1, виходи яких повністю семантично узгоджені з запитаними виходами Out_r в ярусі L_0.

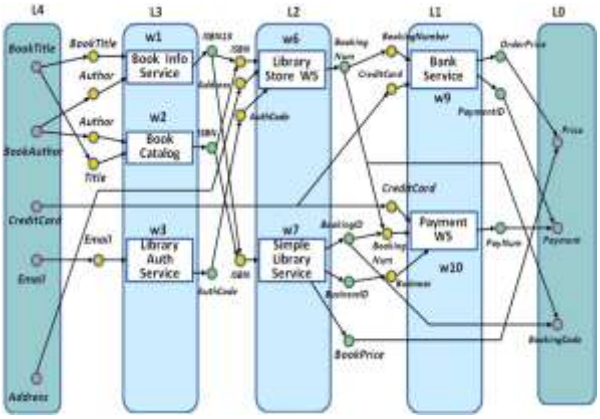


Рис. 4. Граф композиції $w_1 - w_{10}$ "назад"

Другий ярус L_2 складають ті сервіси, виходи яких повністю узгоджені з входами попередніх ярусів, і т. д. Для кожного сервісу алгоритм зберігає ті його виходи, які вже були семантично узгоджені з входами сервісів попереднього ярусу. Він лише встановлює відповідність між новими входами сервісів попереднього ярусу та рештою ще не зіставлених виходів кожного сервісу. Отже, кількість неузгоджених виходів кожного сервісу монотонно зменшується з кожним ярусом.

Як оригінальну композицію "від входів", так і розглянуту композицію "від виходів" можна далі оптимізувати за допомогою оригінальних технік застосування інвертованих індексів, обрізування, домінування інтерфейсу [38] та мінімізації кількості сервісів для заданого рівня певних показників QoS [39].

Для ресурсно ефективної побудови СоПС додатково запроваджено спеціальні класи оптимізованих композицій – нормальні та безнадлишкові [3].

Припустиму композицію семантичних Веб-сервісів названо *нормальною*, якщо в ній немає сервісів, входи яких

не є семантично зіставленими з виходами деякого іншого сервісу чи вхідними концептами запиту In_r (названі *ірелевантними за виходом*) і сервіси, виходи яких не є семантично зіставленими з входами деякого іншого сервісу чи вихідними концептами Out_r (названі *ірелевантними за входом*).

Запропонований алгоритм перетворення допустимої композиції у нормальну, або її *нормалізації*, полягає в послідовному вилученні з неї кожного ірелевантного за входом і за виходом сервісу (як первинного, так і поточно отриманого) разом з усіма семантичними зв'язками його входів з іншими сервісами композиції. Очевидно, алгоритм зберігає припустимість композиції, натомість "очищуючи" її від "даремних" сервісів, як показано, наприклад, на рис. 4.

Нормальну композицію названо *безнадлишковою*, якщо після вилучення з неї довільного сервісу (разом з усіма семантичними зв'язками його входів з іншими сервісами) та нормалізації хоча б один з вихідних концептів запиту Out_r вже не буде семантично зіставлений з виходом деякого сервісу, тобто буде порушено припустимість композиції. Отже, безнадлишкова композиція є мінімальною у тому сенсі, що з неї не можна вилучати сервіси.

Розроблений алгоритм перетворення нормальної композиції у безнадлишкову передбачає поярусне (починаючи, наприклад, з останнього ярусу) виконання для кожного сервісу в ярусі таких кроків.

1. Перевіряння можливості його вилучення без порушення припустимості композиції.

2. За можливості – вилучення сервісу, нормалізація залишкової композиції та розгляд наступного сервісу вже в отриманій нормальній композиції. За неможливості вилучення – перехід до наступного кроку.

3. Аналіз наступного сервісу ярусу, поки всі вони не будуть проаналізовані.

Описаний алгоритм залишає в ярусі лише сервіси, які не можна вилучити, або робить його пустим. Далі кроки 1–3 виконують для ярусу, суміжного з обробленим, доки не буде розглянуто всі яруси.

У залежності від порядку розгляду ярусів і сервісів у ярусі наданий алгоритм генерує різні конфігурації безнадлишкових композицій. Одну з них, отриману після вилучення з нормальної композиції на рис. 4 сервісів w_2, w_7, w_{10}, показано на рис. 5. Для побудови СоПС доцільна подальша мінімізація, на множині безнадлишкових композицій, цільових критеріїв нефункціональної природи, зокрема кількості сервісів, їх сукупної вартості, показників структури їх зв'язків за допомогою технік [42].

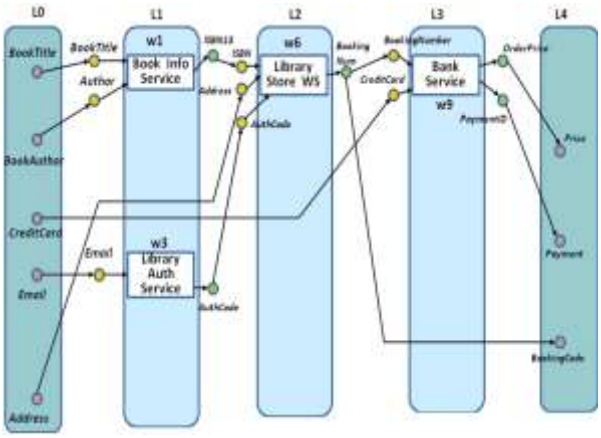


Рис. 5. Безнадлишкова композиція

Побудова адаптивного композитного семантичного сервісу. Як показано на рис. 1, розглянута динамічна композиція семантичних Веб-сервісів (нормальна чи безнадлишкова) ефективно підтримує поточний діловий процес певного споживача. Але в актуальних ПрО, де ролі, взаємодії, потреби й пріоритети суб'єктів і параметри інфраструктури доступу до сервісів різнорідні й непередбачувано змінні, вимоги до якості СоПС дедалі жорсткіші, а ресурси обмеженіші, – цього недостатньо [38]. Стійке досягнення декларованих переваг сервісного програмування – зниження трудо-

місткості та контролювання якості й окупності СоПС [1, 2, 5] – додатково вимагає їх *адаптивності й* застосовності третіми сторонами ПрО для довільного методу динамічного компонування. Адаптивність визначено як здатність до змін поведінки для задоволення нових вимог і пристосування до нових (не)передбачених ситуацій.

Запропоновано підхід до побудови динамічного композитного семантичного Веб-сервісу із зазначеними властивостями, названого адаптивним композитним сервісом (АКС) [3]. Він узагальнює модель адаптивного компонування Веб-сервісів різнорідними споживачами [19] та розглянуті вище формальні засоби забезпечення адаптовності ПС [22] для семантичних Веб-сервісів. Підхід є інваріантним щодо графо-орієнтованого методу компонування складників АКС, фіксуючи єдину природну вимогу: якщо композиції Веб-сервісів з двох різних множин реалізують певний запит, то його реалізує й композиція елементів їх об'єднання. Зокрема, описи графо-орієнтованого алгоритму за входами/виходами (P. Rodríguez-Mier [38]) і нормальної й безнадлишкової композиції засвідчують їх відповідність цій вимозі.

Розвинутий підхід передбачає:

а) формулювання вимог до процесу побудови/супроводу АКС;

б) визначення згідно з ними рамкових моделей: власне АКС; процесу його побудови (з етапами проектування й адаптування до змін потреб споживачів у функціях та умов виконання); окремого етапу; операцій адаптування АКС;

в) уточнення рамкових моделей для ефективних графо-орієнтованих методів компонування, насамперед нормального та безнадлишкового;

г) розвиток методів адаптування АКС за проявів її незадовільної якості.

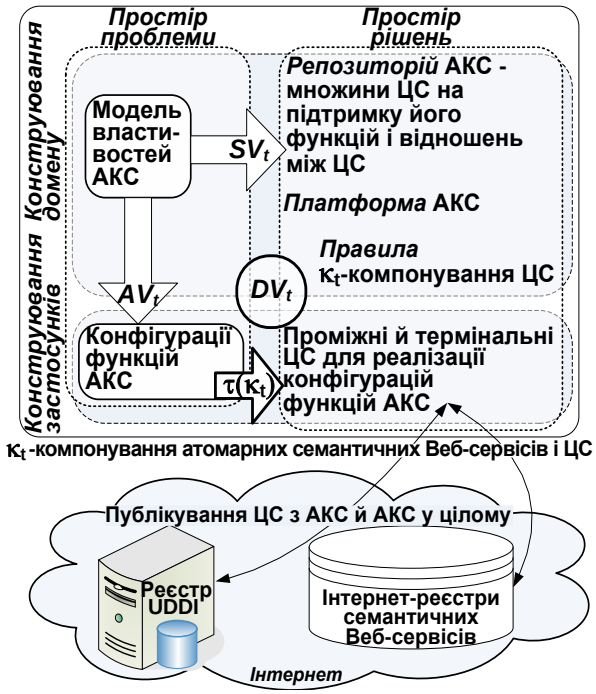
Висунуті вимоги охоплюють: відстеження залежностей між подіями – підставами адаптації АКС, узгодження дій з адаптації на його рівнях (див. рис. 1) та урахування неповноти даних про стан АКС і контекст його виконання.

Сутність підходу – побудова АКС як динамічної лінії змінних композитних семантичних Веб-сервісів, названих *цільовими* (ЦС), на підтримку ділових процесів різнорідних суб'єктів цільової ПрО. Динамічна лінія продуктів [3, 19] є розвитком розглянутої вище статичної лінії за рахунок припущення про змінність у часі складу і структури просторів проблеми й рішень згідно зі змінами потреб споживачів та умов виконання продуктів. Відповідно, МВ стає змінною в часі, а канонічна варіабельність – варіабельністю під час виконання.

Запропоновано моделювання процесу побудови АКС як процесу поетапного керування його варіабельністю під час виконання. Як і для адаптовних ПС, його подано композицією п'яти вищезазначених функцій керування варіабельністю. Однак в АКС їх призначенням є формування нових або зміна наявних ЦС, а їх операції виконуються в інформаційному середовищі поточного етапу між моментами адаптаційної зміни АКС, коли його варіабельність усе ще відповідає вимогам до неї з боку зацікавлених сторін.

Відповідно, визначено дві рамкові підстави адаптування АКС: незадовільність його очікуваної чи поточної варіабельності. Для діагностування типу незадовільності й рекомендування адаптаційних дій з її опрацювання розроблено діагностичну підмодель варіабельності композитних Веб-сервісів [19], що узагальнює об'єктно-компонентну діагностичну модель варіабельності, розглянуту вище.

Наразі побудовано рамкову модель АКС. Це поповнюваний кортеж його узгоджених подань у просторах Проблеми й рішень у періоди між адаптаційними змінами, впродовж яких його варіабельність відповідає потребам зацікавлених сторін. Високорівневу внутрішню структуру такого подання показано на рис. 6 і детально розглянуто в [3] разом з операціями адаптування.



Позначення:
SV_t – модель динамічної варіабельності в структурі АКС;
AV_t – модель динамічної варіабельності в артефактах АКС;
DV_t – діагностична модель варіабельності АКС;
κ_t – метод компонування семантичних Веб-сервісів;
$\tau(\kappa_t)$ – відображення реалізації функцій АКС за рахунок κ_t

Рис. 6. Внутрішня структура АКС у період між його змінеммями в моменти t і t+1

Висновки

Проаналізовано особливості постановки та вирішення загальних проблем конструювання сервіс-орієнтованих прикладних програмних систем на компонентному й сервісному рівнях сервіс-орієнтованої архітектури. Підсумовано актуальні результати фахівців ІПС НАН України з опрацювання цих проблем, отримані в парадигмах компонентного та сервісного програмування, висвітлено взаємозв'язки між ними. Визначено низку актуальних проблем подальшого розвитку розробленого формального апарату побудови сервіс-орієнтованих прикладних програмних систем як адаптивних композитних сервісів у семантичному Веб-середовищі:

1) формалізація показників якості композитного сервісу як підстав для його адаптування;

2) оптимізація динамічних композицій з використанням показників якості в ролі цільових критеріїв і в ролі обмежень;

3) забезпечення ефективної взаємодії різнорідних сервісів з великими обсягами неструктурованих і різнотипних даних, зокрема розвиток підходу Extract-Transform-Load;

4) перехід до семантичної сервіс-орієнтованої й сервіс-компонентної архітектури.

1. Андон П.І., Бабенко Л.П. Проблеми і можливості програмування в середовищі SEMANTIC WEB. *Проблеми програмування*. 2012. № 2-3. С. 363–373.
2. Андон П.І., Дерецький В.О. Проблеми побудови сервіс-орієнтованих прикладних інформаційних систем в semantic web середовищі на основі агентного підходу. *Проблеми програмування.* 2006. № 2-3. С. 493–502.
3. ДР 0112U002764 Розробка теоретичних основ та прикладних питань побудови сервіс-орієнтованих прикладних програмних систем у семантичному Веб-середовищі. Звіт про НДР (заключний). К.:, ІПС НАНУ, 2016. 280 с.
4. Сторінка "W3C Semantic Web Activity". [Електронний ресурс]. – Режим доступу: https://www.w3.org/2001/sw/.
5. Laskey K. OASIS Reference Architecture Foundation for Service Oriented Architecture. Version 1.0. 2012. [Електронний ресурс]. Режим доступу: https://docs.oasis-open.org/ soa- rm/soa-ra/v1.0/soa-ra-v1.0.pdf.
6. Lausen H. A Conceptual and Formal Framework for Semantic Web Services (v1). 2011. [Електронний ресурс]. Режим доступу: https://www.sti-innsbruck.at/sites/default/ files /D2.4.5.pdf.
7. Лаврищева Е.М. Парадигмы программирования сборочного типа в программной инженерии. Сб. трудов международной конф. УкрПРОГ–2014. К.: ИПС НАНУ, 2014. С. 76–92.
8. Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы, технологии и CASE – средства программирования. К.: Наук. думка. 2013. 283 с.

9. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов. 2-изд. Доп. и перераб. К.: Наук. думка, 2009. 372 с.
10. Лаврищева Е.М. Сборочное программирование. Теория и практика. *Кибернетика и системный анализ*. 2009. № 6. С. 1–12.
11. Лавріщева К.М., Колесник А.Л., Стеняшин А.Ю. Об'єктно-компонентне проектування програмних систем. Теоретичні і прикладні питання. Вісник КГУ, серія фіз.-мат. наук. 2013. № 4. С. 150–162.
12. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. М.: Издательство "Бином". СПб.: "Невский диалект", 2001. 560 с.
13. Лаврищева Е.М., Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС. М.: Финансы и статистика, 1982. 127 с.
14. Глушков В.М., Стогний А.А., Лаврищева Е.М., Моренцов Е.И. Система автоматизации производства программ (АПРОП). Киев: Ин-т кибернетики АН УССР, 1976. 134 с.
15. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика. Киев, Наук. думка, 2006. 454 с.
16. Грищенко В.М. Метод об'єктно-компонентного проектування програмних систем. *Проблеми програмування*. 2007. № 2. С. 113–125.
17. Lavrischeva E., Stenyashin A., Kolesnyk A. Object-Component Development of Application and Systems. Theory and Practice. JSEA. 2014. N 7. P. 756–769. [Електронний ресурс]. Режим доступу: http://file.scirp.org/pdf/JSEA_2014080715053066.pdf.
18. Ремарович С. Системи виявлення Web-сервісів в сервіс-орієнтованій архітектурі: проблеми і рішення. Проблеми програмування. 2015. № 3. С. 53–71.
19. Слабоспицька О.О. Технологічна модель процесу побудови та використання адаптивної композиції Web-сервісів. Проблеми програмування. 2015. № 2. С. 52–62.
20. Лавріщева К.М., Коваль Г.І., Бабенко Л.П. та ін. Нові теоретичні засади технології виробництва сімейств програмних систем у контексті генерувального програмування: монографія [Текст]. Ін-т програмних систем. Київ: [б. и.], 2011. 277с. Деп. у ДНТБ України 5.10.11, №67-Ук2011.
21. Лаврищева Е.М., Слабоспицкая О.А. Технология моделирования изменяемых программных продуктов и систем. Труды XII Междунар. науч.-практ. конф. TAAPSD'2015. Киев, 23-26 ноября 2015г. С. 118–127.
22. Slabospickaya O. Feature Model of Software Product Line Enhancing to Enable Product Adaptability. Bull. of Univ. of Kiev. Series: Ph.&Math., spec. is. 2014. P. 151–158.
23. Лавріщева К.М., Слабоспицька О.О., Колесник А.Л., Коваль Г.І. Теоретичні аспекти керування варіабельністю в сімействах програмних систем. Вісник КНУ, серія фіз.-мат. науки. 2011. № 1. С. 151–158.
24. Слабоспицька О.О. Технологічна модель процесу автоматизованого виробництва сімейств програмних систем. *Проблеми програмування*. 2011. № 1. С. 39–48.
25. Kolesnyk A., Slabospitskaya O. Tested Approach for Variability Management Enhancing in Software Product Line. Proc. 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6–10, 2012. [Електронний ресурс]. Режим доступу: ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p–155–162.pdf.
26. Slabospitskaya O., Kolesnik A. The Model for Enhanced Variability Management Process in Software Product Line. Mayr H.C., Kop C., Liddle S., Ginige A. Information Systems: Methods, Models and Applications. Revised selected papers of 4-th International United Information Systems Conference (UNISCON 2012). Yalta, Ukraine, June 2012. P. 162–171.
27. Лаврищева Е.М., Слабоспицкая О.А. Подход к экспертному оцениванию в программной инженерии. *Кибернетика и системный анализ.* 2009. № 4. С. 151–168.
28. Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю. Основы инженерии качества программных систем. 2-е изд. К.: Академпериодика. 2007. 672 с.
29. Лаврищева Е.М, Коваль Г.И., Коротун Т.М. Подход к управлению качеством программных систем обработки данных. *Кибернетика и системный анализ*. 2006. № 5. С. 174–185.
30. Коваль Г.І. Основні задачі підтримки прийняття рішень в інженерії надійності програмних систем. *Проблеми програмування*. 2005. № 3. С. 35–41.
31. Мороз Г.Б., Коваль Г.И., Коротун Т.М. Концепция профилей в инженерии надежности программных систем. *Мат. машини і системи*. 2004. № 1. С. 166–182.
32. Slabospickaya O. A process for consistent and informed assessment of software reliability over its life cycle. *Математичні машини і системи*. 2010. № 1. С. 218–224.

33. Мороз Г.Б., Коротун Т.М. Ризико-операційний підхід до вирішення проблеми оптимального випуску програмних систем. *Проблеми програмування* (Спецвипуск конференції УкрПРОГ-2006). 2006. № 2-3. С. 231–236.
34. Слабоспицька О.О., Коваль Г.І. Інтегрована технологія інтелектуального керування ризиками програмних проектів. *Кибернетика и системный анализ*. 2009. № 6. С. 137–143.
35. 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.
36. Андон Ф.И., Бабко Л.Д. Стандартизация инженерии систем и программных средств в Украине. *Кибернетика и системный анализ.* 2009. Т. 45. № 6. С. 144–148.
37. Лавріщева К.М., Зінькович В.М., Колесник А.Л. Інструментально-технологічний комплекс розробки і навчання прийомам виробництва програмних систем. Свідоцтво про реєстрацію авторського права на твір № 45292 від 27.08.2012. Державна служба інтелектуальної власності України. Київ. 2012.
38. Rodriguez-Mier P., Pedrinaci C., Mucientes M., Lama M. An Integrated Semantic Web Service Discovery and Composition Framework, 2015. [Electronic resourse]. Mode of access: https://arxiv.org/pdf/1502.02840.pdf.
39. Rodriguez-Mier P., Mucientes M., Lama M. Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition. IEEE Transactions on Services Computing, 2015. [Electronic resourse]. Mode of access: https://arxiv.org/pdf/1509.06254.pdf.

References

1. Andon P.I., Babenko L.P. The Problems and Opportunities of Programming in SEMANTIC WEB. Problems in Programming. 2012. N 2-3. P. 363–373. (In Ukrainian).
2. Andon P.I., Deretsky V.A. The Problems of Applied Service-oriented Information Systems Engineering in Semantic Web based on Agent Approach. Problems in Programming. 2006. N 2-3. P. 493–502. (in Ukrainian).
3. DR 0112U002764 Theoretical Fundamentals and Applied Issues Investigation of Applied Service-oriented Information Systems Engineering in Semantic Web. Research report (final).). K.: SSI NASU, 2016. 280 p.
4. Page ”W3C Semantic Web Activity”. [Electronic Resource]. Mode of access: https://www.w3.org/2001/sw/.
5. Laskey K. OASIS Reference Architecture Foundation for Service Oriented Architecture. Version 1.0. 2012. [Electronic Resource]. Mode of access: https://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-v1.0.pdf.
6. Lausen H. A Conceptual and Formal Framework for Semantic Web Services (v1) 2011. [Electronic Resource]. – Mode of access: https://www.sti-innsbruck.at/sites/default/files /D2.4.5.pdf.
7. Lavrischeva E.M. Paradigms of programming assembling type in software engeneering. Problems in Programming. 2014. N 2-3. P. 121–132. (In Ukrainian).
8. Лаврищева Е.М. Software Engineering of Computer Systems. Paradigms, technologies and CASE-tools for Programming. К.: Nauk. Dumka. 2013. 283 p.
9. Lavrischeva E.M., Grischenko V.N. Assembly Programming. Basics of Software Industry. Kyiv: Naukova Dumka, 2009 (2nd ed.). 372 p. (in Russian).
10. Lavrischeva E.M. Assembly Programming. Theory and Practice. Cybernetics and Systems Analysis, 2009. N 6. P. 1–12. (in Russian).
11. Lavrishcheva E., Stenyashin A., Kolesnyk A. Object-Component De-sign. Theoretical and Applied Issues. Visn. Ser. Fiz.-Mat. Nayky, Kyiv St. Univ. im. Tarasa Shevchenka. Special Issue, 2013. N 4. P. 150–162. (in Russian).
12. Booch G. Object-oriented Analysis and Design with Examples of C++ Applications. M.: "Binom Ed.", SPb.: "Nesky dialect", 2001. 560 p.
13. Lavrischeva E.M., Grischenko V.N. Interconnection of Multilingual Modules in OS ES. M.: Finansy i Statystika, 1982. 127 p. (in Russian).
14. Glushkov V.M., Stogniy A.A., Lavrischeva E.M., Morentsov E.I. The System for Program Production Automation (APROP). Kiev:

Institute of Cybernetics of USSR AS, 1976. 134 p. (in Russian).
15. Lavrischeva E.M. Methods of Programming. Theory, Engineering, Practice. Kiev, Nauk. Dumka, 2006. 454 p.
16. Grischenko V.N. The Method for Object-Component Software Design. Problems in Programming. 2007. N 2. P. 113–125. (in Russian).
17. Lavrischeva E., Stenyashin A., Kolesnyk A. Object-Component Development of Application and Systems. Theory and Practice. JSEA. 2014. N 7. P. 756–769. [Electronic Resource]. Mode of access: http://file.scirp.org/pdf/JSEA_2014080715053066.pdf.
18. Remarovich S. The Systems for Web-service Discovery in Service0Oriented Arcitecture: Problems and Solutions. Problems in Programming. 2015. N 3. P. 53–71. (in Ukrainian).
19. Slabospitska O.O. Technological Model for the Process of Adaptive Web Service Composition Engineering and Use. Problems in Programming. 2015. N 2. P. 52–62. (in Ukrainian).
20. Lavrischeva E., Koval G., Babenko L. et al. New Theoretical Foundations of Production Methods of Software Systems in Generative Programming Context. Electronic monograph, in: UK-2011, Vol. 67. Kiev (2011) (in Ukrainian).
21. Lavrischeva K.M., Slabospickaya O.A. Technology for Changeable Software Products and Systems Modelling. Proc. 12th Int. Conf. TAAPSD'2015. Kiev, November 23-26. P. 118–127. (in Russian).
22. Slabospickaya O. Feature Model of Software Product Line Enhancing to Enable Product Adaptability. Bull. of Univ. of Kiev. Series: Ph.&Math., spec. is. 2014. P. 151–158.
23. Lavrischeva K., Slabospitskaya O., Kolesnik A., Koval G. The Theoretical View for Software Family Variability Management. Visn., Ser. Fiz.-Mat. Nayky, Kyiv Univ. im. Tarasa Shevchenka. 2011. N 1. P. 45–53. (in Ukrainian).
24. Slabospickaya O.O. Technological Model for Software Family Automated Production Process. Problems in Programming. 2011. N 1. P. 39–48. (in Ukrainian).
25. Kolesnyk A., Slabospitskaya O. Tested Approach for Variability Management Enhancing in Software Product Line. Proc. 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6–10, 2012. [Electronic Resource]. Mode of access: ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p–155–162.pdf.
26. Slabospitskaya O., Kolesnik A. The Model for Enhanced Variability Management Process in Software Product Line. Mayr H.C., Kop C., Liddle S., Ginige A. Information Systems: Methods, Models and Applications. Revised selected papers of 4-th International United Information Systems Conference (UNISCON 2012). Yalta, Ukraine, June 2012. P. 162–171.
27. Lavrischeva E.M., Slabospitskaya O.O. An approach for Expert Assessment in Software Engineering. Cybernetics and Systems Analysis. 2009. N 4. P.151–168. (in Russian).
28. Andon P.I., Koval G.I., Korotune T.M., Lavrischeva E.M., Suslov V.Yu. The Fundamentals for Software Quality Engineering. 2-nd ed. K.: Akademperiodika. 2007. 672 p. (in Russian).
29. Lavrischeva E.M., Koval G.I., Korotune T.M. An Approach for Quality Management of Data Processing Software Systems. Cybernetics and Systems Analysis. 2006. N 5. P. 174–185. (in Russian).
30. Koval G.I. The Basic Tasks for Decision Making in Software Reliability Engineering. Problems in Programming. 2005. N 3. P. 35–41. (in Ukrainian).
31. Moroz G.B., Koval G.I., Korotune T.M. The Concept of Reliaility Profiles in in Software Reliability Engineering. Mathematical machines and Systems. 2004. N 1. P. 166–182. ((in Russian).
32. Slabospickaya O. A process for consistent and informed assessment of software reliability over its life cycle. Mathematical machines and Systems. 2010. N 1. P. 218–224.
33. Moroz G.B., Korotune T.M. Risk-Operational Approach for the Problem of Optimal Software Delivery Solving. Problems in Programming. 2006. N 2-3. P. 231–236. (in Ukrainian).
34. Slabospitska O.O., Koval G.I. An Integrated Technology for Software Project Risks Smart Management. Cybernetics and Systems Analysis. 2009. N 6. P. 137–143. (In Ukrainian).
35. Lavrischeva E.M., Koval G.I., Korotune T.M. Approaches for Quality Engineering of Software System Families. Problems in Programming. 2008. N 2-3. P. 219–228. (in Ukrainian).

36. Andon P.I., Babko L.D. Systems and Software Engineering Standardization in Ukraine. Cybernetics and Systems Analysis. 2009. Vol. 45. N 6. P. 144–148. (in Russian).
37. Lavrischeva K.M., Zinkovich V.M., Kolesnik A.L. Instrumental-Techno-logical Complex for Software Development and Production Scills Learning. A Certificate for authors' intellectual property N 45292 at 27.08.2012. State Intellectual Property Service of Ukraine. Kyiv, 2012. (In Ukrainian).
38. Rodriguez Mier P., Pedrinaci C., Mucientes M., Lama M. An Integrated Semantic Web Service Discovery and Composition Framework. 2015. [Electronic resourse]. Mode of access: https://arxiv.org/pdf/1502.02840.pdf.
39. Rodriguez-Mier P., Mucientes M., Lama M. Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition. IEEE Transactions on Services Computing, 2015. [Electronic resourse]. Mode of access: https://arxiv.org/pdf/1509.06254.pdf.

Одержано 15.08.2017

Про авторів:

Андон Пилип Іларіонович,
академік НАН України,
директор Інституту програмних систем НАН України,
Кількість наукових публікацій в українських виданнях – 400.
Кількість наукових публікацій в зарубіжних індексованих виданнях – 10.
http://orcid.org/0000-0001-6546-0826.

Слабоспицька Ольга Олександрівна,
кандидат фізико-математичних наук,
старший науковий співробітник.
Кількість наукових публікацій в українських виданнях – понад 50.
Кількість наукових публікацій в зарубіжних індексованих виданнях – 5.
http://orcid.org/0000-0001-6556-0947.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ,
проспект Академіка Глушкова, 40.
Тел.: +38(044) 526 4286.
E-mail: olsips2017@gmail.com