УДК 681.03

ВИЗНАЧЕННЯ ПРЕДМЕТУ – ПРОГРАМНА ІНЖЕНЕРІЯ

Лавріщева К.М.

Інститут програмних систем НАН України,
03187, Київ, проспект Академіка Глушкова, 40.
Тел.: (044) 526 3470, lem@isofts.kiev.ua

Запропоновано нове визначення програмної інженерії, як наукової, інженерної і виробничій дисципліни. Дається визначення базових понять, об'єктів, процесів і змісту цих дисциплін. Наведені теоретичні наробки для об'єктного і компонентного програмування, перевірки правильності, оцінювання якості та менеджменту. Дано аналіз інженерії виробництва цільових об'єктів за компонентами повторного використання.

The article is proposed new interpretations of Software Engineering as science, engineering and practice discipline. The base entity, objects, process and concepts of there discipline are defined. The theories results for objective and component of programming, control of correct, measurement of guiltily and management are described. The engineering of target domain object development from reuse are done.

Посвячується 40-річчу появи терміну "Програмна інженерія".

Вступ

Термін *програмна інженерія* вперше був введений у 1968 р. З того часу протягом 40 років зміст цього поняття, поступово змінювався і вдосконалювався, що обумовлювалося постійним розвитком програмування, спочатку як *мистецтва* одинаків, потім як теоретичної та прикладної *науки*, і, з часом, як *інженерної діяльності*.

В результаті багаторічної праці всесвітнього програмуючого загалу накопичилась велика кількість знань та досвід побудови різноманітних комп'ютерних програм. Вони знайшли відображення у конкретних програмних продуктах широкого застосування та у сукупності теоретичних і прикладних методів і засобів, принципів і правил, а також цілісних процесів виробництва комп'ютерних систем за участю колективів програмістів і інженерів (наприклад, інженерів з якості, оцінювачів програмних продуктів та процесів, тестувальників тощо). В рамках багатогранної діяльності теоретиків та практиків у галузі програмування сформувалися формальні методи доказу, верифікації і тестування програм, математичні моделі надійності, методи оцінювання показників якості програмних продуктів тощо.

Все це обумовило необхідність систематизувати набуті знання і визначити їх у вигляді самостійної предмету (дисципліни) з метою формування спільного бачення проблеми комп'ютерною спільнотою. Спеціально створений комітет спеціалістів з інформатики при ACM і IEEE Computer Society сформував базове ядро знань SWEBOK (Software Engineering body of Knowledge – 2001г.), в якому в концентрованому вигляді подано концептуальний зміст десятьох базових областей (knowledge areas) та дефініція програмної інженерії (ПІ), зокрема [1–4].

Визначення 1. *Програмна інженерія* (Software Engineering) **–** це система методів, способів і дисципліни з планування, розробки, експлуатації і супроводження програмного забезпечення (ПЗ), призначених для промислового виробництва ПЗ. Це інженерна дисципліна охоплює усі аспекти створення ПЗ від початку формулювання системних вимог, через розроблення продукту і до його використання, супроводження та остаточного списання.

На наш погляд ця дефініція дещо звужує сутність не лише поняття предмету ПІ, але й об'єкту, що застосовується у ПІ, тобто програмного забезпечення (ПЗ). Тому спробуємо сформулювати визначення ПІ і її об'єктів у більш широкому розумінні, долучаючи аспекти, що характеризують глибинні питання ПІ, як наукової та інженерної дисципліни.

Відомо, що будь-яка наука – це система перевірених практикою знань, які відображають загальні питання, поняття і закономірності їхнього розвитку. Вона встановлює зв'язки з іншими науками, а також впливає на їхній розвиток. Стосовно програмної інженерії можна сказати, що вона інтегрує в собі принципи математики та головні положення фундаментальних наук, а саме, теорії алгоритмів, математичної логіки, теорії керування, теорії множин, доказу тощо.

Ці науки утворюють теоретичний базис програмної інженерії, необхідний для побудови абстракцій програм за їхніми базовими поняттями та принципами, що перелічені далі по кожній з фундаментальних наук базису:

– в теорії алгоритмів – нормальні алгоритми, обчислювальні функції, машина Тьюрінга, алгоритмічні алгебри, граф-схеми, моделі алгоритмів і програм тощо;

– в математичній логіці – логічні числення і висловлювання теоретичного створення правильних умововиведень формальним шляхом та логіко–алгебраїчний апарат специфікації програм;

– у теорії доказу – математичне доведення за аксіомами і твердженнями програм, вивід теорем, обгрунтування непротирічності і алгоритмічно невирішених проблем, а також теорія верифікації програм (VDM, RAISE, Z, методи Хоара, Дейкстри тощо), теорія надійності ПЗ;

– у теорії множин – квантори загальності, існування, операції над множинами, що застосовуються для формального подання різних сукупностей програмних об'єктів і аксіом;

– у теорії керування – принципи, методи та загальні закони планування і керування процесами отримання і оброблення інформації в кібернетичних і управлінських системах.

Крім цього фундаменту в систему знань програмної інженерії включаються:

– формальні методи програмування – специфікація програм, їхній доказ, верифікація і тестування, а також математичні моделі надійності, ризику тощо;

– прикладні методи, а саме прийоми, принципи, правила, окремі дії і цілісні процеси життєвого циклу (ЖЦ) створення комп'ютерних систем, що є інструментами колективної розробки, застосовуваними виконавцями великих програмних проектів;

– методи керування колективами, а саме планування за мережними графіками, контролювання робіт у процесах ЖЦ, вимірювання і оцінювання якості проміжних результатів виробництва, прогнозування і регулювання строків і вартості виготовлення продукту, а також його сертифікації [5, 6].

Тобто програмна інженерія, як спадкоємиця науки програмування, залучила у сферу своїх знань усі теоретичні і прикладні досягнення, набуті за період її існування, і, таким чином, склалася, як науково-інженерна дисципліна, яка входить до складу комп'ютерної науки (Computer science).

Тому природним буде вважати визначення 1, наведене у ядрі знань SWEBOK, вузьким, і дати нове визначення (визначення 2) програмної інженерії, як наукової і інженерної дисципліни, у ширшому тлумаченні.

Визначення 2. *Програмна інженерія* – розділ комп'ютерної науки, який вивчає методи і засоби побудови комп'ютерних програм; відображає закономірності розвитку та узагальнює накопичений досвід програмування; оперує об'єктами (модулями, компонентами, програмними аспектами тощо) та визначає автоматизовані операції щодо їхнього застосування; виробляє правила, порядок інженерної діяльності і керування технологічним процесом побудови з простих об'єктів нових, більш складних, *цільових* об'єктів (програмного забезпечення, програмних систем (ПС), сімейств систем, програмних проектів тощо), а також методи вимірювання і оцінювання готового продукту.

При тлумаченні програмної інженерії в більш широкому сенсі мається на увазі теорія програмування та інженерія створення програмних продуктів, що сформувалася шляхом адаптації загальних методів керування (наприклад, методу критичного шляху, PERT, за операціями), яким властивий розподіл робіт між різними виконавцями проекту, оцінювання трудовитрат, вартості виготовлення продукту та оцінки якості.

Таким чином, програмна інженерія – це наукова і інженерна дисципліни створення програмних продуктів (рис. 1).

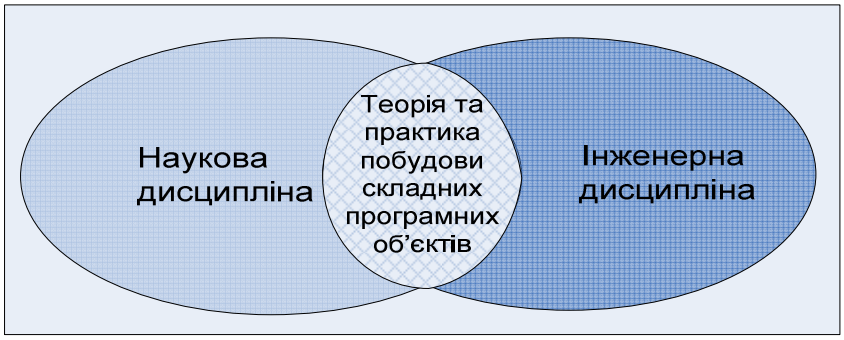

Рис. 1. Наукова і інженерна дисципліни програмної інженерії

На перетині областей визначення ПІ (овалів на рис. 2) – теорія та практика побудови складних програмних об'єктів. *Теорія побудови* – це теорія програмування за абстрактними специфікаціями (графовими і структурними схемами, функціями і композиціями, дескрипторами і номінативними даними, сценаріями використовування (use case діаграми), а також формальна перевірка відповідності об'єктів специфікаціям за методами доказу, верифікації, інспекції тощо. *Практика побудови* – це застосування теоретичних і практичних методів інженерії програмування через використання засобів перевірки (верифікації, валідації, тестування) специфікацій об'єктів, інструментів їхнього послідовного трансформування до результуючого коду та інженерія оцінювання і сертифікації різних показників якості (надійності, продуктивності, ефективності тощо) виготовленого програмного продукту.

Визначення програмної інженерії як наукової дисципліни

На відміну від математичної або інших фундаментальних наук, метою яких є отримання нових знань для вирішення відповідних задач, метою програмної інженерії є застосування знань для розробки складних програмних об'єктів, де знання — це уособлення загальної теорії побудови програм для комп'ютерів, орієнтованої на виготовлення продукту, впровадження якого принесе певну користь користувачеві.

Програмна інженерія як наукова дисципліна – це теоретичні, *формальні* методи та відповідні засоби побудови складних програмних об'єктів. *Побудова* має на меті аналіз предметної області, що автоматизується, і продукування результуючого коду для вирішення задач на комп'ютері. Інтегровані в програмну інженерію фундаментальні науки, вищезгадані (рис.1), а також наука програмування складають її загальний теоретичний фундамент, який надає базові поняття, об'єкти і формальні механізми, необхідні для завдання програмним продуктам (ПП) загальних властивостей та специфічних рис щодо встановлених вимог до них. ПІ як наука включає:

1) основні поняття і об'єкти;

2) теорію програмування і методи керування виготовленням програмного продукту;

3) засоби і інструменти процесів розроблення продукту.

1. Основні поняття програмної інженерії це: дані і їхні структури (прості і складні), функції і композиції, базові об'єкти (модуль, об'єкт, компонент, каркас, контейнер, повторно використовуваний компонент (ПВК) тощо) і цільові об'єкти, що будуються.

Розроблення простих об'єктів виконується через елементарні дії щодо їхнього формального опису, а розроблення цільових об'єктів – через застосування інженерних методів вимірювання і оцінювання, включаючи керування тривалістю і вартістю виробництва.

Дамо загальне визначення цільових об'єктів ПІ.

Програмна система (Application) – комплекс інтегрованих програм і засобів, що реалізують набір взаємопов'язаних функцій деякої предметної області в заданому середовищі. У комплекс можуть входити: прикладні системи (наприклад, програми розрахунку зарплати, обліку матеріалів на складі тощо), загальносистемні програмні засоби (наприклад, транслятор, редактор, СКБД і т.п.), спеціалізовані програмні засоби для реалізації функцій захисту інформації, забезпечення безпеки та ін. *Спосіб виготовлення* – інженерія ПС (application engineering), що включає процеси ЖЦ, методи розробки і процедури керування, а також методи і засоби оцінювання продуктів і процесів з метою їхнього удосконалення.

Програмне забезпечення – сукупність програмних засобів, які реалізують функції комп'ютерної системи (або функції технічної апаратно-програмної системи), включаючи загальносистемні засоби (наприклад, ОС, СКБД, вбудовані підсистеми контролю показників технологічних процесів, оброблення сигналів тощо) та прикладні програмні системи. Так, функціями деякої ОС є керування задачами, програмами, даними і т.п. ОС може входити до складу ПС або бути ідентичною функціям програмної системи. *Спосіб виготовлення* – інженерія розроблення програм для реалізації задач стосовно ПЗ.

Сімейство систем (systems family) – сукупність програмних систем із загальним (незмінним для всіх членів сімейства) і керованим (змінним) набором характеристик, що задовольняють визначеним потребам прикладної області (домену). *Спосіб виготовлення* – інженерія домену (Domain Engineering) або конвеєрне виробництво однотипних ПП за єдиною схемою на основі спеціально розроблених базових членів сімейства й інших готових програмних активів (assets) за допомогою базового процесу або автоматизованої лінійки продукту (Product line).

Програмний проект (program project) – унікальний і інтегрований комплекс взаємозалежних заходів, орієнтованих на досягнення цілей і задач об'єкта розроблення за визначеними вимогами до строків, бюджету та характеристик очікуваних результатів діяльності від нього. *Спосіб виготовлення* – інженерія процесу розроблення і менеджменту проекту.

Складні програмні об'єкти – сукупність взаємопов'язаних цільових об'єктів різних типів (сімейство систем), які виконують необхідні функції в складній системі, подані як самостійно розроблені прості та цільові об'єкти або вибрані з репозитарію готових ресурсів.

2. Теорія програмування – сукупність формальних методів, мов і засобів опису та проектування цільових об'єктів, а також методів їхнього доказу, верифікації і тестування [5–11]. Поряд з об'єктами теорії програмування до програмної інженерії залучені формальні методи керування проектом (персоналом, матеріальними та фінансовими ресурсами) і його окремими характеристиками. Відповідно проведеної нами класифікації методів теорії програмування, у програмній інженерії застосовані наступні (рис. 2):

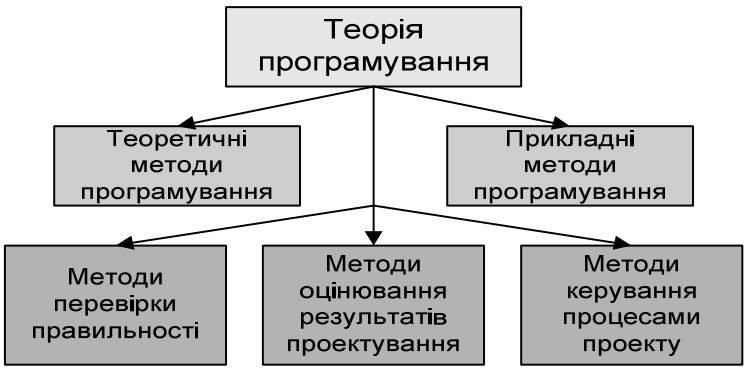

Рис. 2. Сукупність методів програмної інженерії

– методи програмування теоретичні (алгебраїчний, алгоритмічний, експлікативний, функціональний, VDM, RAISE тощо) і прикладні (об'єктний, компонентний, аспектний, генеруючий тощо), призначені для проектування різних типів цільових об'єктів;

– методи перевірки правильності за формальними процедурами (твердження, вивід, доказ) та тестуванням ПП;

– методи оцінювання результатів послідовного проектування (проміжних робочих продуктів) і кінцевого продукту відносно встановлених показників (надійність, якість, точність, продуктивність тощо);

– методи керування (менеджменту) і контролю розроблення проміжних результатів під час виконання процесів проекту, а також допоміжні розрахункові методи (трудовитрат кожного розробника, вартості робіт тощо).

В Інституті програмних систем НАН України отримані наукові результати, які відносяться до теорії програмування. Розглянемо їх.

Створення теорії об'єктного і компонентного програмування. Одним з напрямків сучасного програмування, для якого створено теорію, є об'єктне і компонентне програмування. Вони є спадкоємцями модульного програмування. Їхня загальна особливість – виробництво з готових "деталей" або ресурсів складних програмних систем.

Об'єктну теорію побудовано з використанням формалізму трикутника Фреге (рис. 3) для узагальнення поняття об'єкту, як елементу реального світу з відповідними властивостями і характеристиками (**знак** – ідентифікатор, **денотат –** сутність знаку, **концепт** – семантика денотату), які визначаються на рівнях об'єктного аналізу із залученням математичних формалізмів їхнього опису та уточнення рис. 3 відрізняючи один об'єкт від іншого.

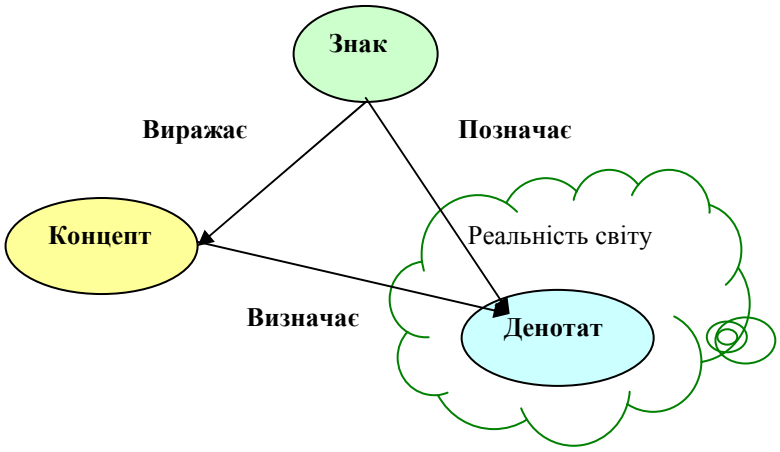

Рис. 3. Трикутник Фреге

Сутність теорії об'єктного аналізу

Визначено алгебру аналізу $\Sigma = (E', \Psi, P)$, де $E'=(E_1, E_2, E_n)$ – множина об'єктів, $\Psi = \{decds, decdn, comds, comdn, conexp, connar\}$ – операції над елементами E' і $P=(P_1, P_2, ..., Pr)$ – множина предикатів для завдання концептів об'єктів. Кожна з операцій має певний пріоритет та арність, орієнтована на зміну денотатів однорідних та неоднорідних об'єктів, а також на розширення та звуження їхніх концептів. Доведено, що множина операцій Ψ алгебри Σ є повною системою операцій об'єктного аналізу і функцій деталізації, екземпляризації, агрегації.

Результатом об'єктного аналізу ПрО є сукупність об'єктів зі своїми властивостями і характеристиками. Визначені об'єкти за компонентним методом відображаються у програмні реалізації, а саме, компоненти (методи) і інтерфейси, які містять операції звернення одного до іншого. Встановлюється тісний теоретичний і практичний зв'язок між об'єктним аналізом і компонентним методом створення програмних систем.

Компонентний метод програмування розширений формальними моделями і операціями:

– зовнішній і внутрішній компонентної алгебри;

– перетворення даних різномовних компонентів, що об'єднуються в складні системи.

Компонентні моделі це:

– модель компонента Comp = (*CName, CInt, CFact, CImp, CServ*),

де *CName* – унікальне ім'я компоненту; $CInt = \{CInt^i\}$ – множина інтерфейсів, пов'язаних з компонентом; *Cfact* – інтерфейс керування екземплярами компоненту; $CImp = \{CImp^j\}$ – множина реалізацій компоненту; $CServ = \{CServ^r\}$ – множина системних сервісів. Між компонентами можливі такі відносини, як наслідування, екземпляризації, контракту, зв'язування та взаємодії.

– модель інтерфейсу $CInt^i = (IntName^i , IntFunc^i , IntSpec^i)$,

де $IntName^i$ - ім'я інтерфейсу; $IntFunc^i$ – функціональність (сукупність методів); $IntSpec^i$ – специфікація інтерфейсу (типів, констант, інших елементів даних, сигнатур методів і т.д.).

– модель компонентного середовища CE = (*NameSpace, IntRep, ImpRep, CServ, CServImp*),

де *NameSpace* = {*CName*^{m}} – множина імен компонентів середовища; IntRep = {IntRepi} – репозитарій інтерфейсів компонентів середовища; *ImpRep* = {*ImpRep*^{j}} – репозитарій реалізацій; *CServ* = {*CServ*^{r}} – інтерфейс множини системних сервісів; *CServImp* = {*CServImp*^{r}} – множина реалізацій для системних сервісів.

Алгебра компонентного програмування це:

Зовнішня алгебра: $\Psi = \{ CSet, CESet, \Omega 1\}$,

де *CSet* – множина компонентів Comp, CESet – середовище E з множин компонентів і інтерфейсів, розташованих у його репозитаріях, $\Omega 1 = \{CE_1, CE_2, CE_3, CE_4\}$ – операції алгебри, до яких віднесені операції оброблення компонентів: інсталяції (розгортання) $CE_2 = Comp \oplus CE_1$, об'єднання компонентних середовищ $CE_3 = CE_1 \cup CE_2$, видалення компонента з компонентного середовища $CE_2 = CE_1 \setminus Comp$, заміщення компоненти $Comp_1$ на $Comp_2$ – $CE_2 = Comp_2 \oplus (CE_1 \setminus Comp_1)$.

Внутрішня алгебра: $\varphi = \{ CSet, CESet, \Omega 2\}$,

де $\Omega 2 = \{O_{refac}, O_{Reing}, O_{Rever}\}$ – сукупність операцій еволюції, а саме:

– $O_{refac} = \{AddOImp, AddNImp, ReplImp, AddInt\}$ – рефакторінгу,

– $O_{reing} = \{rewrite, restruc, adop, supp, conver\}$ – реінженерії,

– $O_{Rever} = \{visual, metric, restruc, design, rewrite\}$ – операції реверсної інженерії.

Тобто, компонентна алгебра $\Xi = \{\Psi \cap \varphi\} = \{ CSet, CESet, \Omega 1\} \cap \{ CSet, CESet, \Omega 2\} = \{CSet, CESet, \Omega 1, \Omega 2\}$ – це множина компонентів та операцій обробки, еволюції та взаємодії компонентів з компонентним середовищем.

Сутність перетворення типів даних компонованих компонентів

Для формалізованого опису типів даних різномовних компонентів використаний алгебраїчний підхід, в якому кожному типу даних ставиться у відповідність алгебраїчна система з множиною значень та операцій над об'єктами даного типу. Кожній операції перетворення типів даних відповідає ізоморфне відображення однієї алгебраїчної системи в іншу.

У класі простих типів даних $t = b$ (*bool*), *з* (*char*), i (*int*), r (*real*)) і *с*кладних $t = a$ (*array*), z (*record*), u (*union*), e (*enum*) мов програмування (МП) побудовано два класу алгебраїчних систем їхнього перетворення:

$\Sigma_1 = \{G_\alpha^b, G_\alpha^c, G_\alpha^i, G_\alpha^r\}$,

$\Sigma_2 = \{G_\alpha^a, G_\alpha^z, G_\alpha^u, G_\alpha^e\}$.

Кожен елемент класу визначається на множині значень цих типів даних та операцій над ними:

$G_\alpha^t = <X_\alpha^t, \Omega_\alpha^t>$,

де $t = b, c, i, r, a, z, u, e$ – тип даних, X_α^t – множина значень, які можуть приймати змінні цього типу даних, Ω_α^t – множина операцій над цими типами даних.

Відображення G_α^t в G_β^q для мов l_t та l_q $L = \{l_\alpha\}_{\alpha=1, n}$ і операцій $t \rightarrow q$ з множини $T_\alpha = \{T_t\}$ є ізоморфізмом**,** якщо

– типи даних t, q, визначені на одній і тій же множині;

– операції Ω_α^t та Ω_β^q вживають різні типи даних для значень X_α^t та X_β^q .

– $\Omega = \Omega_\alpha^t \cap \Omega_\beta^q$ не порожні для двох систем $G_\alpha^{t'} = < X_\alpha^t, \Omega >$ і $G_\beta^{q'} = < X_\beta^q, \Omega >$ (коли t є рядок, а тип q – речовинне, то між X_α^t та X_β^q нема ізоморфізму),

– $|G_\alpha^t| = |G_\beta^q|$ рівні, лінійно впорядковані і зберігають лінійний порядок елементів для мов l_t і l_q .

Перетворення між масивами й записами зводяться до ізоморфізму між відповідними алгебраїчними системами шляхом структуризації даних, операцій селектора і конструювання. Механізми перетворення типів даних різномовних компонентів доведені теоремами [5] і є основою доказу правильності зв'язків пар компонентів е різних мовах програмування (МП).

Іншим напрямком забезпечення таких зв'язків є розроблений стандарт ISO/IEC 11404–96, в якому подано мову LI (Language Independent) незалежно від МП. Але ця мова не отримала інструментальної підтримки до наступного часу. Тому користувачі вибирають найбільш відповідну практичну реалізацію інтерфейсу в різних середовищах. Прикладом сучасного практичного перетворення є настанови [8] варіантів модулів–посередників у класі сучасних мов: C/C++, Visual C++, Visual Basic, Matlab, Smalltalk, Lava, LabView, Perl, що створюються неавтоматизованим засобом.

Запропонована теорія об'єктного, компонентного програмування та перетворення типів даних різномовних компонентів використовується в інженерії виробництва складних програм з готових компонентів повторного використання (ПВК).

Наробки щодо перевірки правильності компонентів. Перевірка правильності програм – це формальні специфікації, доказ, верифікація і тестування. *Специфікація* – це формальний опис функцій і даних програм, з якими ці функції оперують. На ній базуються методи доказу програм з залученням математичного апарату для завдання правильного рішення у вигляді аксіом, тверджень, передумов і пост-умови, як попередніх і заключних правил одержання результату. *Верифікація і валідація* – це перевірки правильності виконання функцій програм у відповідності з заданими вимогами замовника. *Тестування* – це головний напрямок наших досліджень, як метод виявлення при виконанні вихідного коду за тестовими даними різних помилок, дефектів, відмовлень і збоїв, викликаних нерегулярними ситуаціями або аварійним припиненням роботи

системи. Усі особливості сучасного аналізу і тестування програм запропоновані в [9, 10, 13]. Головними чинниками тестування є ризик, інтенсивність відмов та визначення часу тестування в залежності від них.

В цьому напрямі було розроблено [9]:

– математичну модель визначення оптимального часу тестування компонентів ПС з урахуванням ризиків відмов під час експлуатації ПС та метод оцінювання ризику відмов компонентів;

– розрахунок оптимального часу тестування t_e^*, щоб прибуток був максимальним і обчислювався за формулою $K(t_0 |t_e) = \Delta R\ (t_0 |t_e) – C(t_e)= C_м\ (\mu(\ t_0) - \mu(\ t_0 + t_e) + \mu(\ t_e)) - \ c_1\ t_e - c_2\mu(t_e)$, а її похідна $K(t_0 |t_e)$ по t_e дорівнює $K'(t_0 |t_e) = C_м\ (\lambda(t_e) - \lambda(\ t_0 + t_e)\ ) - c_1 - c_2\lambda(t_e)$ в залежності від функції зростання надійності, функції інтенсивності $\lambda(t)$ і ризику $C_м$;

– розрахунок часу виконання компонента з урахуванням ризиків компонентів.

Отримані результати – є основа стратегії тестування, орієнтованої на визначення тих компонентів, які потребують більш тривалого часу тестування за умови, що вартість (трудомісткість) їхнього тестування відповідає величині ризику використання компонента при роботі системи. Стратегія входить як головна діяльність в створений базовий процес тестування, який упорядковує задачі тестування, що виконуються на різних підпроцесах розроблення системи з компонентів. Він включає також розподіл обов'язків між учасниками процесу, стандарти для опису документів, метрики процесу, критерії початку та завершення процесу. Запропонований підхід до тестування захищений у дисертації, пройшов перевірку в конкретних проектах і забезпечив отримання більш якісного продукту

Методи оцінювання продуктів. Дослідження задач отримання програмних продуктів з гарантованою якістю виконувалися протягом багатьох років (1992–2005) в межах науково-дослідних проектів ДКНТ, Міністерства науці та НАН України, а також стандарту ISO/IEC 12207 про ЖЦ, гармонізованого в ІПС НАН України, та ДСТУ 9126 про якість програмного продукту, який охоплює аспекти проектування, вимірювання та оцінювання показників якості на всіх процесах ЖЦ та кінцевого результату. Крім того, системно досліджені багато чисельні публікації, ядро знань програмної інженерії – SWEBOK і ядро знань керування проектами – PMBOK (Project Management of knowledge), розроблені відповідно міжнародним комітетом спеціалістів у області інформатики та Інститутом керуванняпроектами (США). В результаті цих досліджень розроблено власний і формальний погляд на проблему якості, який відображений у ряді методик за всіма наведеними аспектами вимірювання, досягнення та оцінювання показників якості. Вони пройшли перевірку на задачах замовника (Міноборони України) і системно викладені у монографії [13], яка отримала другий передрук у 2007р. і має великий попит у СНГ.

Оригінальні результати інженерії якості программного продукта такі:

– модель якості, орієнтовану на оцінку надійності;

– модель розподілу надійності системи з компонентів, основаної на функції $Q_{nc} = \sum_{j=1}^{l} v_j^* \cdot q_j = \sum_{s=1}^{m} w_s^* \cdot r_s$ корисності системи в залежності від вагомих коефіцієнтів w_s і надійності $q_j = \prod_{n \in E_j} r_n$ окремих компонентів;

– концептуальна модель прийняття рішень щодо керування якістю, включаючи методи систематичного контролю надійності, починаючи з ранніх стадій ЖЦ, вимірювання кількісних вимог за надійності компонентів і прогнозування дефектів.

Ці результати захищені у кандидатської дисертації [11] і мають подальшій розвиток щодо застосування у генеруючому і мовному програмуванні стосовно доменів.

Методи керування проектом. Менеджмент проекту – це керівництво роботами команди виконавців програмного проекту для його реалізації як продукту з використанням загальних методів управляння, планування й контролю робіт (бачення майбутнього продукту, стартові операції, планування ітерацій, моніторинг і звітність), керування ризиками і конфігурацією, а також ефективною організацією команди виконавців проекту. Менеджмент проекту залежить від масштабу проекту або "змісту і межі проекту" [15, 16].

Масштаб проекту – це сукупність мети проекту та запланованих витрат часу і різних засобів. Тобто це своєрідний тривимірний простір (мета-час-гроші), в якому живуть учасники проекту та і сам проект. Для проекту складається «трійне обмеження» – зміст проекту, термін і вартість, які враховуються під час узгодження різноманітних вимог до розроблення проекту. Якість виконання проекту залежить від рівноваги цих трьох факторів, за координацію й реалізацію яких відповідає менеджер проекту, а за ідейну, функціональну сторону проекту – головний фахівець проекту.

Наукові результати щодо даного напрямку такі:

– формальна модель керування проектуванням інформаційних систем, що враховує матеріальні, фінансові та трудові ресурси, необхідні для виконання розробки проекту системи;

– метод формування варіанту плану X робот проекту в вигляді сіткового графіка В, включаючи: послідовність робіт ($l_i \in L$), їхній обсяг q_i і вид W_i, ресурси $R = < R_L, R_S >$ і норми їхнього використання ($NR_i \in NR$), закон розподілу випадкових величин $F = \{F_1,..., F_r\}$, часу t в плановому періоді $[t_0,T]$ і ймовірність закінчення робот з урахуванням критерію для оптимального плану $K(X^*)=min\ K(X)$;

– нові принципи моделювання задач керування технологічним процесом проектування системи з урахуванням базових параметрів – масштабу, вартості, часу, а також цілей та наявності необхідних ресурсів.

Ці результати захищені у дисертації [15], а також пройшли апробацію у конкретних системах автоматизації освітніх процесів (наприклад, „документообіг в освіті України", портали „Діти України", „Вчитель новатор" 2004-2007), при викладанні менеджменту проекту в вищих навчальних закладах Академії педагогічних наук та захищені в дисертації. Надалі проводяться роботи по застосуванню розроблених методів менеджменту в інших інформаційних системах освіти України.

3. Засоби і інструменти ПІ. До об'єктів, що є артефактами створюваного програмного продукту (ПП), включаються різного роду описи: вимоги до розробки ПП, погоджені з замовником, архітектура, структури даних, специфікації програм і т.п. Проектування об'єктів виконується за допомогою сучасних візуальних мов, наприклад UML, мов програмування (C++, Java, Pascal тощо) з використанням відповідних інструментальних середовищ, що містять необхідні мовні перетворювачі й інструменти підтримки різних артефактів ПП, що розробляються. В якості засобів їхнього проектування застосовуються діаграми використання, потоків даних, класів, поведінки, а також шаблони, каркаси, темплейти тощо.

Перевірка правильності цих об'єктів здійснюється за допомогою зазначених методів і відповідних інструментів, пристосованих до цілей розроблення різних задач проекту в середовищі проектування. Готовий продукт перевіряється щодо відповідності реалізованих функцій заданим вимогам, тестується за спеціальними методиками, а також піддається вимірюванню та оцінюванню на предмет отримання показників якості, точності, відмовостійкості, захищеності тощо. У середовищі проектування цільових об'єктів застосовуються передові сучасні технології і відповідні інструментально-технологічні пакети інструментів (наприклад, RUP, MSF, Rational Rose, Microsoft Visual Studio тощо). В них міститься не тільки інструменти проектування різних типів цільових об'єктів проектів, а також засоби і інструменти керування проектом, зокрема персоналом, планами і якістю продуктів.

Засоби і інструменти забезпечують автоматизовану підтримку базового процесу виготовлення програмного продукту в організації-розробнику. Класифікацію загальних інструментів, рекомендованих для застосування стосовно всіх видів об'єктів у процесах ЖЦ, подано в ядрі знань SWEBOK.

Визначення програмної інженерії як інженерної дисципліни

Програмна інженерія як інженерна дисципліна (або інженерія) – це сукупність прийомів виконання діяльності, пов'язаної з виготовленням програмного продукту для різних видів цільових об'єктів із застосуванням методів, засобів і інструментів наукової складової програмної інженерії [8–12]. Основу інженерії складають наступні базові елементи процесу виготовлення програмного продукту (рис. 4):

1) ядро знань SWEBOK, як набір теоретичних концепцій і формальних визначень стосовно методів і засобів розроблення та керування програмними проектами, які можуть застосовуватися у інженерії програмування;

2) базовий процес ПІ, як стрижень процесній діяльності в організації-розробнику ПП;

3) стандарти, як набір регламентованих правил конструювання проміжних артефактів у процесах ЖЦ;

4) інфраструктура – умови середовища та методичне забезпечення базового процесу ПІ і підтримка дій його виконавців, що займаються виробленням програмного продукту;

5) менеджмент проекту (PMBOK) – ядро знань з керування промисловими проектами, як набір

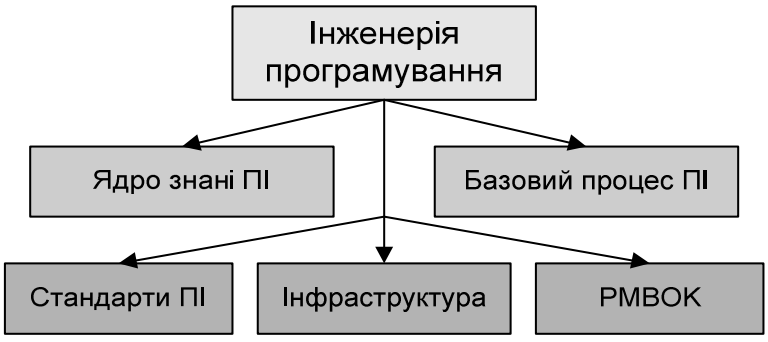

Рис. 4. Базові складові інженерної дисципліни

стандартних процесів, а також принципів і методів планування і контролювання роботами в проекті.

З інженерної точки зору в програмній інженерії вирішуються задачі виготовлення ПП, подані як технологічні процеси формування вимог, проектування і супроводу продукту, а також перевірки операцій базового процесу на правильність виконання різних функціональних задач проекту та вкладання робіт за проектом у заданий замовником строк.

Програмну інженерію можна розглядати з двох пов'язаних точок зору:

– як інженерну діяльність, у якій інженери різних категорій виконують роботи в рамках проекту, використовуючи відповідні теоретичні методи і засоби ПІ, що рекомендовані у ядрі знань SWEBOK, а також стандарти процесів проектування цільових об'єктів за обраними методами;

– як систему керування проектом, якістю і ризиками з залученням правил і положень стандартів ЖЦ, якості та менеджменту проекту.

Інженерна діяльність обов'язково планується та ґрунтується на розподілі робіт у проекті між різними категоріями виконавців. Менеджер проекту – це головна діюча особа проекту, відповідальна за проектування і контроль виконання робіт спеціальними службами інфраструктури проекту в організації, зокрема служби верифікації, тестування, якості тощо. Продукт колективного виготовлення передається замовнику для супроводу. В ньому можуть бути знайдені різні помилки і недоліки, які усувають розробники.

Ця діяльність у програмній інженерії практично вже відпрацьована і за своєю сутністю близька до інженерної діяльності у промисловості, де *інженерія* – це спосіб застосування наукових результатів у виготовленні технічних виробів на основі технологічних правил і процедур, методик виміру, оцінки і сертифікації в цілях задовольняння і отримання користі від виготовленого продукту або товару.

Далі (у 1 – 5) подано загальну характеристику базових елементів інженерної дисципліни виготовлення програмного продукту, показаних на рис. 4.

1. Ядро знань SWEBOK – стислий опис концептуальних основ програмної інженерії. Структурно поділяється на 10 розділів (knowledge areas), які умовно можна розкласти за двома категоріями: проектування продукту і інженерна діяльність. Перша категорія – це методи і засоби розробки (формування вимог, проектування, конструювання, тестування, супровід), друга категорія – методи керування проектом, конфігурацією і якістю та базовим процесом організації-розробника.

Методи ядра знань програмної інженерії менеджер проекту зіставляє з відповідними стандартними процесами ЖЦ, виконання яких забезпечує послідовне розроблення програмного продукту. Наповнення базового процесу програмної інженерії методами з ядра знань SWEBOK, а також задачами і діями стандартного ЖЦ, обумовлює його пристосування до потреб конкретної організації-розробника щодо певної регламентованої послідовності розробки і супроводу програмного продукту. Все це створює технологічний базис інженерії виготовлення конкретного продукту (або низки однотипних продуктів) в організації. На початкових стадіях розробки виконуються процеси визначення вимог до продукту, проектні рішення і каркас (абстрактна архітектура) майбутнього продукту. На основі вимог і каркасу розробляються або вибираються готові прості об'єкти для „наповнення" каркасу змістом для подальшого його доведення до стану готового продукту.

2. Базовий процес (БП) – це метарівень для забезпечення «процесного продукування» продукту. Він містить основні поняття стосовно оснастки, організаційної структури колективу розроблювачів та методології оцінки, виміру, керування змінами і удосконалювання самого процесу. В цілому базовий процес містить множину логічно пов'язаних з ним видів інженерної діяльності організації-розробника та набір засобів і інструментів щодо виготовлення програмного продукту.

3. Інфраструктура – це набір технічних, технологічних, програмних (методичних) та людських ресурсів організації–розробника, необхідних для виконання підпроцесів базового процесу програмної інженерії, орієнтованого на виконання договору з замовником програмного проекту. До технічних ресурсів відносяться: комп'ютери, пристрої (принтери, сканери тощо), сервери і т.п. До програмних – загальносистемне ПЗ середовища розробки, напрацювання колективу, оформлені у вигляді повторно використовуваних компонентів та інформаційне забезпечення. Технологічні та методичні ресурси складають – методики, процедури, правила, рекомендації стандартів щодо процесу і керування персоналом, включаючи комплект документів, що встановлює регламент виконання і регулювання процесів ЖЦ, пристосованих для вирішення конкретних задач проекту. Людські ресурси – це групи розробників та служб керування проектом, планами, якістю, ризиком, конфігурацією та перевірки правильності виконання проекту розробниками.

Засоби, проміжні результати розроблення за процесами ЖЦ, а також методики керування різними ресурсами, виконання БП і застосування методів програмування, зберігаються у базі знань проекту (рис. 5).

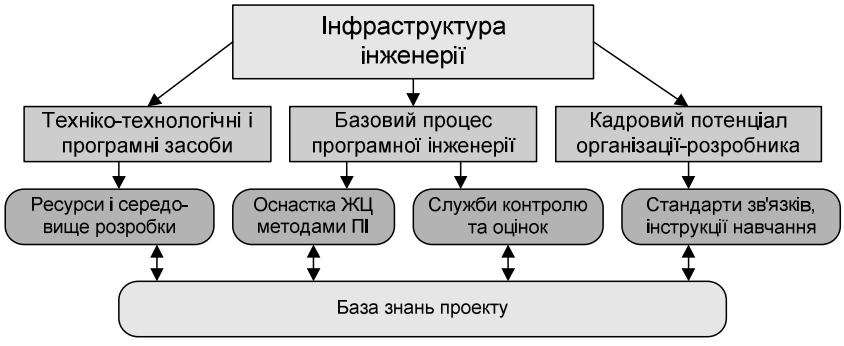

Рис. 5. Загальна інфраструктура проекту

Після виконання проекту і отримання досвіду побудови конкретного продукту, базовий процес і його окремі елементи, подані на даному рис. 5, можуть удосконалюватися (через доопрацювання або зміни прийомів, залучення доробка, змінювання, додавання нових засобів) відповідно до вимог стандарту ДСТУ ISO/IEC 15504-7 (Оцінювання процесів ЖЦ ПЗ. Настанови з удосконалення процесу) з метою підвищення рівня можливостей і оцінки потужності процесу.

Готовність всіх видів забезпечення організації-розробника продуктів, досконаліть виконуваних процесів і набута якість створеного в ній продукту надають підстави для оцінки зрілості організації або сертифікації процесів виробництва ПЗ. Для оцінювання зрілості може застосовуватися модель зрілості CMM (Capability Maturity Models), запропонована Інститутом програмної інженерії SEI США, або інша модель, наприклад, Bootstrap, Trillium тощо. Модель CMM встановлює рівні зрілості організації стосовно створення програмних продуктів. Рівень зрілості визначається наявністю в організації базового процесу, всіх необхідних видів ресурсів (у тому числі і фінансових), відповідних стандартів і методик, а також професіональних здібностей (зрілості) членів колективу організації, здатних виготовляти програмні продукти в заданий строк і встановленої вартості. Модель CMM пропонує п'ять рівнів зрілості, від першого – найнижчого, до п'ятого – найвищого. П'ятий рівень зрілості процесів організації свідчить про здатність команди розробників створювати якісний програмний продукт.

4. Стандарти ПІ – встановлюють технологічно відпрацьований набір процесів зі строго визначеним і регламентованим порядком проведення різних видів робіт у програмної інженерії, зв'язаних з розробленням програмного продукту і оцінюванням його якості, ризику тощо. Стандарти у галузі програмної інженерії регламентують різні напрямки діяльності щодо програмування програмних продуктів. Вони стандартизують термінологію і поняття, життєвий цикл, якість, вимірювання, оцінювання продуктів і процесів. Найбільш важливими серед них є стандарт ISO/IEC 12207 „Процеси життєвого циклу програмного забезпечення" (та його дещо застарілий вітчизняний еквівалент ДСТУ 3918-99), серія стандартів ДСТУ ISO/IEC 14598 "Оцінювання програмного продукту", стандарт ДСТУ ISO 15939 "Процес вимірювання", серія стандартів ДСТУ ISO/IEC 15504 "Оцінювання процесів ЖЦ ПЗ", базові стандарти з якості - ДСТУ ISO 9001 «Системи управління якістю. Вимоги», ДСТУ 2844–94, ДСТУ 2850–94, що регламентують різні аспекти забезпечення якості ПП. Серед стандартів, що безпосередньо пов'язані з якістю ПЗ, слід також назвати проект нової серії стандартів ДСТУ ISO/IEC TR 9126 "Програмна інженерія. Якість продукту" (введення у дію якого заплановано у 2008 році).

У цих стандартах узагальнені знання спеціалістів з технології проектування і інженерних методів керування розробкою, починаючи від встановлення вимог, і закінчуючи оцінюванням якості продукту і можливою його подальшою сертифікацією. Процеси ЖЦ в стандарті ISO/IEC 12207 подають загальні положення, задачі й регламентовані дії по проектуванню, а також рекомендації щодо застосування цих процесів для розроблення і контролю проміжних результатів. У стандарті містяться також організаційні процеси – планування, керування і супроводу. *Процес планування* призначений для складання планів, графіків робіт щодо виконання проекту і розподілу робіт між різними категоріями фахівців, а також для контролю планів і виконаних робіт. *Процес керування проектом* визначає задачі та дії з керування роботами у проекті, виконуваними фахівцями, які володіють теорією керування, а також стеження за плановими строками, що надані замовником проекту. *Процес супроводу* – включає дії щодо покращення готового продукту, виявлення й усунення знайдених в ньому недоліків і внесення нових або видалення деяких функцій у продукті.

Ядро знань SWEBOK і стандарти ЖЦ мають зв'язок. Процесам ЖЦ зіставляються необхідні методи ядра і тим самим визначається базовий процес проекту, що доповнюється методиками і обмеженнями щодо вироблення продукту. Діючі фундаментальні моделі ЖЦ (водоспадна, спіральна тощо), які широко використовуються на практиці, пропонують вкладений в них стиль проектування і реалізації деяких видів продуктів.

5. Менеджмент проекту – це керування розробленням проекту з використанням теорії керування та процесів ядра знань **PMBOK** (Project Management body of knowledge) [9]. В настанові з використання PMBOK подано положення і правила керування часовим виробничим циклом побудови унікального продукту в рамках проекту. PMBOK є стандартом, що розроблений американським Інститутом управління проектами (www.pmi.org), з початку без урахування рівня комп'ютеризації промисловості (1987р.)**,** а потом і з його врахуванням (2000 р.). Слід зазначити, що на теперішній час настанови до PMBOK та SWEBOK введені в статус стандартів, а саме: ISO/IEC TR 19759 ("Guide to the Software Engineering Body of Knowledge (SWEBOK)) та IEEE Std.1490 "IEEE Guide adoption of PMI Standard. A Guide to the Project Management Body of Knowledge), російській варіант PMBOK можна знайти за адресою в Інтернет (http://petukhov.zaklad.ru/rmbok2004_rus.pdf).

Ядро знань PMBOK містить опис лексики, структури процесів і областей знань, відображаючи сучасну практику керування проектами в різних областях промисловості. В ньому визначені процеси ЖЦ проекту і головні області знань, згруповані за задачами: ініціація, планування, використання, моніторинг і керування, завершення. Крім того, область знань – інтеграція визначає прийняття рішень про використання ресурсів в кожний момент виконання проекту і керування загальними задачами проекту.

Область знань *керування вмістом проекту* включає процеси, які необхідні для виконання робіт за проектом (виключаючи надлишкові дії), а також для його планування з розбивкою робіт на більш прості для спрощення процесу керування. Область *керування якістю* містить процеси і операції досягнення цілей проекту щодо якості, правила і процедури для поліпшення процесу досягнення цілей і забезпечення якості відповідно заданим вимогам, а також контролю результату на відповідність стандартам якості. Область *керування*

людськими ресурсами організації і розподілу робіт між виконавцями відповідно до їхньої кваліфікації і професіоналізму містить процедури регламентування виконання робіт з розроблення програмного продукту.

Таким чином, стандарт PMBOK, як і стандарт ISO/IEC 12207, мають багато спільного, особливо стосовно організаційних процесів з керування проектом і інженерією доменів. Нагадаємо, що *інженерія домену* – це процес і набір робіт щодо побудови систем сімейства із застосуванням компонентів повторного використання, програмних застосувань (Applications) та систем домену на інженерній основі. ПВК є інженерним ресурсом, який може використовуватися багаторазово і давати значну економію у виробництві нових продуктів. На даний час розроблені механізми формалізованого опису об'єктів і компонентів, їхніх інтерфейсів, а також методи їхнього розміщення у репозитаріях компонентів і інтерфейсів проектів.

Ядро знань SWEBOK і PMBOK пов'язані подібними моделями ЖЦ, методами й інструментами керування процесами виконання проекту. Проведений розгляд питань вироблення програмних продуктів на процесній і інженерній основі демонструє те, що сформульовані змістовні грані фундаменту ПІ (рис. 6).

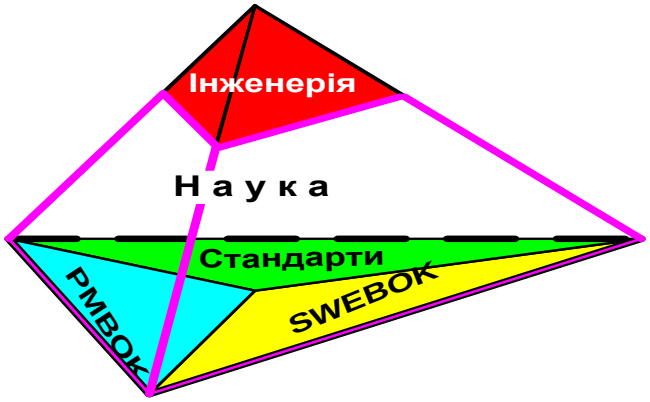

Рис. 6. Базові грані програмної інженерії

Грані SWEBOK, СТАНДАРТИ, PMBOK відображають виробничій фундамент, котрий застосовується при проектуванні програмних систем з використанням наукових і інженерних досягнень у програмної інженерії. Вони забезпечують технологічність та досягнення якості розроблення програмних продуктів різного призначення.

Програмна інженерія в виробничому вимірі

Загальне призначення програмної інженерії – це практика, тобто конкретна побудова комп'ютерних програм, систем і інструментів із застосуванням теоретичних і інженерних методів ПІ.

Головна особливість виробничого застосування у наступному часі – це використання розроблених готових програм і інформаційних ресурсів Інтернет (MatLab, Greenstone, Grid-системи й ін.). Доступ до них може здійснити будь-який користувач і одержувати безкоштовно або на комерційній основі готовий програмний ресурс, як сервіс. Він може бути одноразово використаний для рішення відповідної задачі, або як деяка програма постійного і багаторазового застосування в деякому домену. Сьогодні сформувалися три інженерні підходи щодо застосування таких готових ресурсів: reusing engineering, application engineering, domain engineering. Вони використовують як готові ресурси повторно використовувані компоненти (ПВК), додатки (застосування) і системи. Застосування готових ресурсів, як багаторазово використаного готового продукту, дає значну економію при виробництві з них нових програмних систем і сімейств систем. Усі види ПВК зберігаються в сховищах проекту – репозитаріях [3–7,18].

Інженерія ПВК – це систематична і цілеспрямована діяльність для підбора реалізованих і представлених у репозитарії ПВК. Система проектується з них нагору. Спочатку створюється загальна структура – каркас продукту, дається його опис і за цим описом готові компоненти і ПВК інтегруються в систему. Модель специфікації компонента має такий вигляд:

$$\text{ПВК} = (T, I, F, R, S),$$

де T – тип компоненту, I – множина інтерфейсів компонента; F – функціональність компонента; R – реалізація – програмний код; S – сервіс для взаємодії з середовищем або набір правил розгортання.

Кожний з елементів специфікації компонента це видима або прихована від користувача частина його абстракції. Залежно від складності ПВК їх можна розділити на такі групи:

– прості компоненти (функція, модуль, клас, ПВК, та ін.);

– об'єкти-компоненти, що мають інтерфейс, функцію і реалізацію на будь-якій МП, а також можливості доповнювати специфікації шаблоном розгортання та інтеграції;

– готові прості ПВК (наприклад, beans компоненти в Java, AWT компоненти, класи тощо);

– складні ПВК типу каркасів, патернів з елементами групування декількох простих ПВК в закінчену функціональність і їхню взаємодію між собою при рішенні деякої загальної задачі системи.

Готові ПВК при розміщенні у репозитарії мають метаінформацію відносно інтерфейсів, які реалізують компоненти; механізмів опису і середовища повторного використання. Технологія їхнього застосування базується на таких особливостях:

– відображення здатності ПВК надавати свої можливості під час компіляції або динамічно під час виконання;

– стандартизований опис для зручного аналізу і розуміння його іншою особою, крім розробника;

– здатність забезпечувати зміни, не зачіпаючи цільову спрямованість і додавання нових параметрів, а також ре факторинг – трансформацію компонента із збереженням функціональності, але з можливою зміною структури і початкового коду;

– збереження параметрів конфігурації (шаблонів налагодження) в постійній пам'яті для використання в певний час;

– реєстрація повідомлень про події, одержані від інших об'єктів через посилання, повідомлення, а також групування компонентів в файли для подальшого повторного використання;

– застосування ПВК в різних мовних середовищах;

– адаптація ПВК до різних контекстів їхнього застосування і виділення властивостей, які заважають повторному використанню і модифікації в конкретних цілях.

Згідно стандарту ISO/IEC 12207 діяльність щодо інженерії ПВК класифікується як організаційна і планована інженерна діяльність, яка полягає у виявленні загальних і специфічних рис компонентів для прийняття рішень про їхню використання в розробці нових ПС [13, 18].

Відповідно каталогу репозитарію знаходяться необхідні ПВК, які можна з'єднати в нову програмну конструкцію. Саме повторне використання дає можливість розглядати її як систематичну і цілеспрямовану діяльність, яка базується на двох процесах ЖЦ.

Перший процес – це створення ПВК шляхом:

– вивчення спектру вирішуваних задач ПрО, виявлення серед них загальних властивостей і функцій;

– побудови компонентів, що реалізують виявлені функції у вигляді ПВК;

– розробка каталогу для зберігання виготовлених компонентів і організації пошуку необхідних компонентів по запитах користувачів.

Другий процес – конструювання нових систем з готових компонентів шляхом:

– визначення цілей і вимог до системи;

– пошуку в каталозі репозитарія ПВК, що відповідають вимогам до новій системи;

– інтеграція ПВК із забезпеченням інтерфейсу з підсистемами та іншими компонентами.

Перший процес це вкладення капіталу, другий – отримання прибутку за рахунок заощадження трудовитрат від застосування готових ПВК. Інвестиції в повторне використання вимагають оцінки ефективності вкладення капіталу, прогнозування термінів і обсягів повернення цього вкладення, оцінки ризиків та ін. Бізнес повторного використання як будь-який бізнес вимагає спеціальних умов щодо менеджменту всієї інженерної діяльності за ПВК. Критерії успіху такого бізнесу визначаються меншими трудовитратами та зусиллями, ніж розробка нових разових продуктів.

До компонентів ПВК висуваються такі вимоги, як незалежність від конкретної платформи, наявність стандартного інтерфейсу і параметрів настроювання на нове середовище, можливість їхній взаємодії в системі без внесення в них змін.

Розробці ПС за допомогою ПВК відповідає модель ЖЦ з такими загальними етапами:

– аналіз об'єктів і відносин ПрО, яка реалізується, для виявлення ПВК, що мають загальні властивості, притаманні групам об'єктів цієї області;

– адаптація наявних в базі репозитарія ПВК, розробка нових функціональних компонентів, не представлених в цій базі і доведення їх до рівня ПВК;

– розробка інтерфейсів компонентів і їхнього розміщення в репозитарії інтерфейсів системи;

– інтеграція ПВК з урахуванням інтерфейсів з іншими елементами створюваної системи і формування конфігурації цієї системи.

Повторні компоненти можуть бути прикладними і загальносистемними. *Прикладні компоненти* виконують окремі завдання і функції прикладної області діяльності домену (домени бізнесу, комерція, економіка і т.п.), які можуть використовуватися надалі як готові в якості прикладних систем в інших доменах з аналогічними функціями.

До *загальносистемних компонентів* належать компоненти загального і універсальні призначення (транслятори, редактори тестів, системи генерації, інтеграції, завантажувачі та ін.), а також загальносистемні сервісні засоби, які забезпечують системне обслуговування і надають різні види сервісів для багатьох створюваних програмних систем різного призначення. Вони використовуються всіма прикладними системами в процесі нього проектування і виконання. Універсальні системні компоненти (ОС, СКБД, мережне забезпечення, електронна пошта та ін.) забезпечують функціонування будь-яких компонентів, обмін даними і передачу повідомлень між усіма видами систем, розташованих в різних середовищах і платформах комп'ютерів.

Зв'язок між прикладними і загальносистемними засобами здійснюється через стандартні інтерфейси, що забезпечують взаємодію різних типів компонентів через механізми передачі даних і повідомлень.

Інженерія застосувань. Ця інженерія ґрунтуються на багаторазовому використанні ПВК і різних програмних елементів для отримання продукту, задовольняючому вимогам до цього застосування.

Проектування одиночних тобто унікальних програм (застосувань) це програмування з ПВК, де конкретні програми складаються з ресурсів..

Інженерія застосування з компонентів містить процеси створення системи і їхній менеджмент. Процес створення починається з аналізу предметної області, задачі якої автоматизуються, побудови концептуальної моделі, на основі якої створюється компонентна модель, що включає проектні рішення по композиції компонентів, використанню різних типів шаблонів, зв'язків між ними й операцій розгортання ПС у середовищі функціонування. Менеджмент – це розподіл робіт за кожним учасником процесу, створення графіку цих робіт для керованого контролю їхнім виконанням відповідно заданого строку і оцінювання якості як окремого компонента, так і їхній сукупності.

Побудова застосування виконується за наступними етапами ЖЦ.

1. *Пошук, вибір ПВК* або розробка нових компонентів, виходячи із системи класифікації компонентів і їхні каталогізації, формалізоване визначення специфікацій інтерфейсів, поводження і функціональності компонентів, а також їхнього анотування і розміщення в репозитарії системи або в Інтернет.

2. *Розробка вимог* (Requirements) до ПС – це формування й опис функціональних, не функціональних і інших властивостей ПС.

3. *Аналіз поведінки* (Behavioral Analysis) ПС полягає у визначенні функцій системи, деталей проектування і методів їхнього виконання, кожний з яких буде створювати відповідну поведінку.

4. *Специфікація інтерфейсів і взаємодій компонентів* (Interface and Interaction Specification) відбиває розподіл ролей компонентів, інтерфейсів, їхню ідентифікацію і взаємодію компонентів через потокі дій (workflow).

5. *Інтеграція набору компонентів і ПВК* (Application Assembly and Component Reuse) у єдине середовище ґрунтується на підборі й адаптації ПВК, визначенні сукупності правил, умов інтеграції і побудові конфігурації каркаса системи.

6. *Тестування компонентів і середовища* (Component Testing) ґрунтується на методах верифікації і тестування для перевірки правильності як окремих компонентів і ПІК, так і інтегрованої з компонентів ПС.

7. *Розгортання* (*System Deployment*) включає оптимізацію плану компонентної конфігурації з урахуванням середовища, розгорнення окремих компонентів і створення цільової компонентної конфігурації для функціонування ПС.

8. *Супроводу ПС* (System Support and Maintenance) складається з аналізу помилок і відмовлень при функціонуванні ПС, пошуку і виправлення помилок, повторного її тестування й адаптації нових компонентів до вимог і умов інтегрованого середовища.

Цей процес є ітераційним. В ньому можливо повертатися на попередні у випадку невиконання деякого правила або обмеження у вимогах.

На кожному з названих етапів процесу проводиться менеджмент, тобто перевіряється правильність його виконання і відповідність вимогам, що сформульовані для системи. На останньому процесі виявлена деяка кількість помилок може привести до повтору п. 2 або 1, коли з'явилися нові обставини у замовника.

Інженерія ПрО. Призначена для побудови систем сімейства з урахуванням області задач домену, загальних і змінюваних характеристик представників сімейства в моделі характеристик. Обрані ПВК або одиночні програмні застосування вбудовуються в нові члени сімейства ПС зі сховищ споруджуваного домену, наприклад, репозитарія.

Технологія розробки сімейства програм включає три види базових процесів [5, 16, 17]:

– розробка ПрО і унікальних, одиночних програм;

– інженерія повторного використання ресурсів;

– менеджмент домену.

Розробка ПрО відноситься до конвеєрної з повторним використанням компонентів. Для ПрО планується створення базових ресурсів, що можуть повторно використовуватися: компоненти, генератори, DSL-описи, моделі аналізу, ПВК, документація й ін. Розробка предметної області – це більш складний виробничий процес, містить такі загальні етапи: аналіз, проектування і впровадження в ПрО одиночних програм або ПВК.

Аналіз області містить у собі аналіз усього сімейства, яке треба побудувати, визначаючи в ньому загальні і відмінні риси і створюючи структурні і поведінкові специфікації сімейства. Аналіз ПрО починається з вибору вимог і їхньої специфікації для системи і членів сімейства. Специфікація вимог – це вхідні дані для ручного або автоматичного створення домену з готових ресурсів.

Впровадження або реалізація ресурсів для використання ПВК в ПрО включає аналіз готових і одиночних компонентів, генераторів, DSL–описів й ін. Сукупність цих компонентів може бути частково або цілком автоматизоване за допомогою генераторів або конфігураторів готових ресурсів. Сгенерировані продукти сімейства можуть містити не програмні артефакти, наприклад, інструкції з користування DSL, домену й ін. Головною поставною частиною побудови домену є м*енеджмент*, що призначений для керування конвеєрною розробкою з повторним використанням ресурсів, включає планування і контроль підбору типових ресурсів, їхню оцінку і перевірку задоволення вимогам. У задачу менеджменту входить також перевірка застосування готових ресурсів для реалізації специфіки ПрО і програмування компонентів простору задач відповідно потреб клієнтів домену.

Таким чином, процес інженерії Про включає:

– аналіз ПрО, побудова структури, виявлення об'єктів і відносин між ними;

– визначення області дій об'єктів ПрО;

– визначення загальних функціональних і змінюваних характеристик, побудова моделі характеристик ПрО з встановленням залежності між різними членами сімейства;

– створення конкретних прикладних членів сімейства з механізмами мінливості незалежно від засобів їхньої реалізації;

– підбор і підготовка компонентів багаторазового застосування для реалізації задач ПрО;

– генерація окремих членів сімейства або домену в цілому.

Етапи цієї схеми забезпечують специфікацію моделі ПрО і її характеристик з простору проблем. Вони трансформуються в архітектуру системи й опис її компонентів. При цьому виконаний підбор готових компонентів забезпечує рішення задач ПрО у відповідному просторі.

Лінійки продуктів. Технологічні прийоми виробництва програмних продуктів з готових компонентів і програмних систем втілені в, так звані, *лінійки продуктів* (Product Line Practice), що відповідають конвеєрному виробництву програмних продуктів на ринковій основі [5].

Даний принцип інженерії відповідає виробництву систем з компонентів ПВК, програм і ПС, які задовольняють специфічним потребам деякого ринку програмної продукції і показникам якості. Лінійка програмних продуктів за фреймворком є підтримкою інженерії ПрО, в завдання якої входить застосування програмних продуктів, вироблених на лінійці продуктів. В межах даного напрямку виробництва досліджується ринок і аналіз потреб покупців, будується виробничий план, процеси і організація їхньої взаємодії. На основі аналізу потреб ринку будується технологічна лінія продукту, в яку включаються методи розробки, тестування і оцінки процесів і продуктів лінії. Інфраструктура побудови лінійки продуктів наведена на рис.7, яка підтримується відповідними керівними матеріалами і методиками.

Рис. 7. Інфраструктура побудови лінійки продукту

Побудова конкретної лінійки для розробки програмного продукту для певного замовника включає:

– обмеження, властиві продуктам лінійки;

– зразки, каркаси, ПВК, які можуть використовуватися на лінії;

– виробничі обмеження, стратегії і методи;

– засоби й інструменти для розробки продукту на лінії.

На основі цих даних будується план створення продукту на лінійці, який враховує терміни, вартість і вимоги до управління виробництвом продукту шляхом:

– контролю плану робіт і відстеження ходу побудови продукту;

– виявлення ризиків і керування ними в процесі виконавської діяльності на процесі проектування продукту;

– прогнозування вартісних і технічних ресурсів;

– застосування технології керування конфігурацією;

– вимірювання і оцінки якості побудованого продукту.

Практичні інструменти цього виробництва – ядро знань SWEBOK, PMBOK і стандарти, в яких подано опис процесу доменної інженерії (Domain engineering process), як нового процесу в моделі процесів ЖЦ. Згідно цього стандарту процес інженерії доменів охоплює ряд видів діяльності: а*наліз домену* (виявлення зв'язків, постійних і змінних вимог, основних понять та моделей); п*роектування домену* (Domain design) тобто каркасу для ПВК, активів і інтерфейсів, узгоджених з моделлю домену); *технологія інженерії домену* з під процесами формування ресурсів (asset provision), бази ресурсів (asset-based development) та супроводу ресурсів шляхом модифікації і еволюції моделі за рахунок готових ресурсів типу ПВК. В результаті застосування технології інженерії ПрО в софтверній організації створюватиметься, підтримуватиметься і розвиватиметься *архітектурний базис* з множини ПВК репозитарію та специфікації загальних і специфічних особливостей різних сторін діяльності доменів.

Загальний висновок. Наука й інженерія, SWEBOK, СТАНДАРТИ, PMBOK як головні елементи програмної інженерії, зв'язані між собою процесами ЖЦ, теорією, методами проектування і керування розробкою проекту. Вони застосовуються при виробництві, як основні положення технології проектування програмних продуктів у певному середовищі розроблення. На жаль, в Україні немає своїх програмних середовищ для практичного розроблення програмних продуктів на інструментальної основі. Цю прогалину заповнюють закордоні інструментальні середовища та системи підтримки програмування, а саме системи програмування з різних сучасних мов (C++, C#., Basic, Pascal тощо). Наприклад, система Express C#, Basic. Вона дає можливості налагодити певну програму в цих мовах до кінцевого результату.

Прикладом реалізації інженерної дисципліни колективного проектування цільових об'єктів є система Visual Studio Teams Systems фірми Microsoft. В ній реалізовано технологію проектування, кодування,

тестування та впровадження проектів. В середовищі системи реалізовано *технологію* підбору, розподілу і виконання робіт між різними групами спеціалістів програмного проекту. Вони природно мають різні рівні знань і навичок в області програмування і тому поділяються на чотири категорії. Кожний спеціаліст з цих категорій отримує роботу відповідно до своїх здібностей, починаючи з підготовчої категорії. Перехід в іншу більш високу кваліфікаційну категорію залежить від якості виконання спеціалістом попереднього завдання і отриманого ним досвіду, як в частині знання проблематики і методів їхнього вирішення, так і самого пакета інструментів фірми. Спеціалісти, що входять до четвертої категорії, несуть відповідальність за правильність розроблення продукту в цілому в певному середовищі.

Наведемо окремі нові задачі, які є перспективними для розвитку програмної інженерії на наступні десятиріччя:

– 15 річний міжнародний проект – теорія і практика верифікації усіх видів продуктів та їхнє накопичення у Інтернет бібліотеках [19, 20];

– узагальнення технологічних засобів побудови проектів передових міжнародних фірм для підняття теоретичного подання процесу виробництва в них;

– розроблення теоретичного і прикладного підгрунття мовно-орієнтованому програмуванню (специфікації специфіки доменів);

– комп'ютеризація математичних, логіко–алгебраїчних та обчислювальних знань тощо.

Автор висловлюю щиру подяку всім, хто приймав участь у розгляді, обговоренні та уточненні предмету програмної інженерії, особлива вдячність своїм колегам Г.І.Коваль та Т.М.Коротун.

1. *Jacobson I.* Object–oriented Software Engineering. A use case Driven Approach, Revised Printing.– New York: Addison Wesley, Publ. Co.– 1994. – 529 p.
2. *Андон Ф.И., Лаврищева Е.М.* Методы инженерии распределенных компьютерных приложений. – Киев: Наук. думка, 1998. – 228 с.
3. *Бабенко Л.П., Лавріщева К.М.* Основи програмної інженерії. Посібник.– К.: Знання, 2001.– 269с.
4. *Соммервил И.* Инженерия программного обеспечения.– Изд. дом „Вильямс", Москва+ Санкт–Петербург+ Киев. – 2002. – 623 с.
5. *Лаврищева Е.М.* Методы программирования. Теория, инженерия, практика. – К.: Наукова думка, 2006.–450с.
6. *Грищенко В.Н., Лаврищева Е.М.* Методы и средства компонентного програмування // Кибернетика и системный анализ, 2003.– №1– С. 39–55.
7. *Грищенко В.М.* Метод об'єктно–компонентного проектування програмних систем // Проблеми програмування. – 2007. – №2. – С. 113–125.
8. *Бей И.*Взаємодія різномовних програм. – Изд.дом „Вильмс". Москва– С.Петербург–Киев, 2005.–.868с.
9. *Коротун Т.М.* Модели и методы инженерии тестирования программных систем в условиях ограниченных ресурсов.–Автореф. диссер. Киев.– Инст. кибернетики им. Академика Глушкова. – 2005. – 21 с.
10. *Мороз Г.Б., Лаврищева Е.М.* Модели роста надежности программного обеспечения.– Киев.–Препринт 92–38, 1992.– 23с.
11. *Коваль Г.И.* Модели и методы инженерии качества программных систем на ранних стадиях жизненного цикла.– Автореф. канд. диссер. – К.: Инст. кибернетики им.академика Глушкова. – 2005. – 19 с.
12. *Лаврищева Е.М., Коваль Г.И., Коротун ТМ.* Подход к управлению качеством программных систем обработки данных // Кибернетика и системный анализ. – 2006. – № 5. – С.174–185.
13. *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.
14. *Лаврищева Е.М., Рожнов А.М..* Концепция аналитической оценки характеристик качества программных компонентов // Проблемы программирования.– Киев, 2004, N 1–2. – С.180–187.
15. *Задорожная Н.Т.* Управляемое проектирование документооборота в управляющих информационных системах. – Автореф. канд. диссерт.–Киев,– Ин–т кибернетики им. Глушкова, 2004. – 21 с.
16. *Задорожна Н.Т., Лавріщева К.М.* Менеджмент документообігу в інформаційних системах освіти. – К.: Педагогічна думка, 2007. – 220 с.
17. *Лавріщева К.М.* Програмна інженерія.–К.: BHV, 2007. – 413с.(у друку).
18. *Бабенко Л.П.* Проблемы повторного использования в программной инженерии// Кибернетика и системный анализ, 1999. – №2. – С.155–166.
19. *Вудкок Д.* Первые шаги к решению проблемы верификации программ. Открытые системы, 2006. – №8. – С. 36–43.
20. *Hoare T., Misra J.* Verified software: Theories, Tools, Experiments. Vision of Grant Challenge project. – Microsoft Research Ltd and the University of Texas at Austin, 2005. – 43 с.