**Лекція 4. Методи, засоби та інструменти програмної інженерії**

Загальне визначення цільових об’єктів програмної інженерії (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 програмних систем (ПС) (application engineering), що включає процеси життєвого циклу (ЖЦ), методи розробки і процедури керування, а також методи і засоби оцінювання продуктів і процесів з метою їхнього удосконалення.

***Програмне забезпечення*** (ПЗ) – 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 прикладних програм (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.

Термін «програмна інженерія» вперше був застосований у 1968 р. на конференції з цієї проблематики NATO Software Engineering Conference.

**Програмна інженерія**, або **інжиніринг програмного забезпечення**, (Software Engineering) – систематичний, дисциплінований, вимірюваний підхід до розроблення використання та підтримки програм. Тим самим було визначено дисципліну, яка мала пришвидшити створення програмного забезпечення вищої якості, дешевшого за вартістю та легшого у супроводі.

У своєму становленні програмна інженерія пройшла **три етапи**.

**До 1970-х років** процес створення програм розглядався як творчий процес, що не підлягає контролю і не може бути стандартизований.

**Характерні риси класичного підходу**: неформальні методи, брак стандартизації і мінімум документації, яку розробляли після завершення створення системи.

**На початку 1970-х років** почали застосовувати **методи управління проектами на базі моделей життєвого циклу інформаційних систем** (далі – ІС), що забезпечило деяку впорядкованість розроблення ПЗ. Проект розбивають на кілька стадій, кожна з яких має чіткі цілі і завдання, результати виконання яких можна перевірити, і забезпечує підгрунтя переходу до подальших робіт. Виконання кожної стадії супроводжується оформленням відповідної документації.

**Проблеми використання методів життєвого циклу**:

– недостатній ступінь формалізації, що породжує неузгодженість елементів ПЗ;

– відсутність ефективного супроводу розроблення і модернізації ІС.

**Наприкінці 1970-х – початку 1980-х рр**. з’явились **методи програмної інженерії**, які поділяють на такі **категорії**:

**евристичні методи**, що включають неформалізовані підходи. Це структурно-орієнтовані (функціонально-орієнтовані), об’єктно-орієнтовані, зорієнтовані на дані, та зорієнтовані на область застосування методи;

**формальні методи**, обгрунтовані математично. До цієї категорії відносять мови специфікацій, зорієнтовані на модель, властивості або поведінку; підходи до уточнення специфікацій (трансформації у кінцевий результат, максимально близький до бажаного – виконуваний програмний продукт); підходи підтвердження точності характеристик (початкових припущень та отримуваного продукту) із використанням теорем і перевіркою точності моделей;

**методи прототипування**, що включають стилі прототипування (створення тимчасових прототипів, еволюційне прототипування, розроблення виконуваних специфікацій); техніки оцінювання результатів прототипування.

Деталізуємо ці методи.

*Формальні методи* засновані на формальних специфікаціях, аналізі, доведенні і верифікації програм. Специфікація записується мовою, синтаксис і семантика якої визначені формально і засновані на математичних концепціях (алгебрі, теорії множин, логіці). Розрізняються наступні категорії формальних методів:

* *мови і нотації специфікації* (specification languages and notations), орієнтовані на модель, властивості і поведінку;
* *уточнення специфікації* (refinement specification) шляхом трансформації в кінцевий результат, близький до кінцевого програмного продукту, що виконується;
* *методи верифікації/доведення* (verification/proving properties), що використовують твердження (теореми), перед- і постумови, формально описуються і застосовуються для встановлення правильності специфікації програм.

Термін *«формальні методи»* має на увазі ряд операцій, до складу яких входять створення формальної специфікації системи, аналіз і доказ специфікації, реалізація системи на основі перетворення формальної специфікації до програм і верифікація програм. У формальній мові системної специфікації закладені математичні концепції. При цьому використовується область дискретної математики, заснованої на алгебрі, теорії множин та алгебри логіки.

Формальні математичні специфікації не містять деталей реалізації системи, але повинні представляти її повну математичну модель. Існує два основні підходи до розробки формальної специфікації:

* алгебраїчний підхід, при якому система описується в термінах операцій і їх відносин;
* підхід, орієнтований на моделювання, при якому модель системи будується з використанням математичних конструкцій, таких, як множини і послідовності, а системні операції визначаються тим, як вони змінюють стан системи.

У 1980-х рр. багато дослідників вважали, що формальні специфікації і формальні методи є найбільш ефективним шляхом підвищення якості ПЗ, строгий і детальний аналіз, який є невід'ємною частиною формальних методів, призведе до створення програм з малою кількістю помилок. Експерти передбачали, що до XXI сторіччя більшість ПЗ буде розроблятися з використанням формальних методів. Але за 25 років формальні методи розробки програмних систем не отримали широкого визнання, незважаючи на те, що при їх використанні може бути досягнуто підвищення якості програм шляхом докази їх правильності. Багато компаній, що розробляють ПЗ, не вважають економічно вигідним застосування цих методів в процесі розробки.

Розвиток структурних методів, управління конфігурацією і т.д. дозволило підвищити якість програм при більш низьких витратах в порівнянні з вартістю розробки формалізованими методами. Крім того, з формальної специфікації погано узгоджуються методи швидкої розробки ПЗ. *В даний час головним критерієм програмної індустрії для деяких класів систем є не якість, а час поставки їх на ринок.*

Формальні методи виявилися рентабельні в обмеженій області застосування: це розробка критичних систем, де важливі такі властивості, як безпека, безвідмовність і захищеність. Прикладами критичних систем, при розробці яких успішно застосовувалися формальні методи, є інформаційні системи управління повітряним транспортом, системи сигналізації на залізниці, бортові системи космічних кораблів і медичні системи управління.

*Евристичні методи* містять у собі:

* структурні методи, засновані на функціональній парадигмі;
* методи, орієнтовані на структури даних, якими маніпулює ПЗ;
* об’єктно-орієнтовані методи, що розглядають предметну область як колекцію об'єктів;
* методи, орієнтовані на конкретну область застосування, наприклад, на системи реального часу, безпеки та ін.

У 1960-1970-і рр. було розроблено багато *евристичних методів,* які допомагають впоратися зі зростаючою складністю програм. Загальною стратегією боротьби зі складністю програмування системи стала її *декомпозиція*, тобто поділ на дрібні складові частини. Розробники перестали обмежуватися єдиним атрибутом якості програм - коректним функціонуванням, і перейшли до структурування.

На основі аналізу потоків даних, діаграм відносин «сутність-зв'язок», інформаційної закритості і ряду інших принципів і методик сформувалися нові проектні методології. Найбільшого поширення набуло *структурне проектування по методу «зверху вниз»*. Метод був безпосередньо заснований на топології традиційних мов високого рівня типу FORTRAN або COBOL. У цих мовах основний базовою одиницею є підпрограма, і програма в цілому приймає форму дерева, в якому одні підпрограми в процесі роботи викликають інші підпрограми. Структурний проектування використовує саме такий підхід: алгоритмічна декомпозиція застосовується для розбиття великого завдання на більш дрібні.

У більшості методів, що з'явилися згодом, пропонувалося безліч формалізованих нотацій і нормативних настанов для проектування програмного забезпечення, були усунені очевидні недоліки структурного проектування.

Соммервілл пропонує розділити евристичні методи розробки програмного забезпечення на три основні групи]:

* • метод структурного аналізу і проектування;
* • метод потоків даних Джексона (JSD - Jackson System Development);
* • об'єктно-орієнтовані методи.

*Методологія структурного аналізу і проектування* ПЗ визначає кроки роботи, які повинні бути виконані, їх послідовність, правила розподілу і призначення операцій та методів. Структурні методології жорстко регламентують фази аналізу вимог і проектування специфікацій і відображають підхід до розробки ПЗ з позицій рецептів "кулінарної книги". Більшість методологій базуються на наступній "класичній" сукупності:

- діаграми потоків даних, що забезпечують аналіз вимог і функціональне проектування ІС;

- розширення для проектування систем реального часу, засновані на діаграмах переходів станів, таблицях рішень, картах і схемах потоків керування;

- діаграми "сутність-зв'язок " для проектування структур даних, схем БД, форматів файлів як частини всього проекту;

- структурні карти Джексона і/або Константайна для проектування міжмодульних взаємодій і внутрішньої структури модулів.

У структурному аналізі і проектуванні використовуються різні моделі, що описуютьo функціональну структуру системи. Визначається ієрархія діаграм потоків даних, що описують асинхронний процес перетворення інформації від її введення в систему до видачі споживачеві. Розробка ПЗ ґрунтується на моделі ВХІД-ОБРОБКА-ВИХІД: дані входять в систему, обробляються або перетворюються і виходять з системи. Така модель використовується у всіх структурних методологіях. При цьому важливий порядок побудови моделі. Традиційний процедурно-орієнтований підхід регламентує первинність проектування функціональних компонент по відношенню до проектуванню структур даних: вимоги до даних розкриваються через функціональні вимоги. При підході, який орієнтується на дані, вхід і вихід є найбільш важливими – структури даних визначаються першими, а процедурні компоненти є похідними від даних.

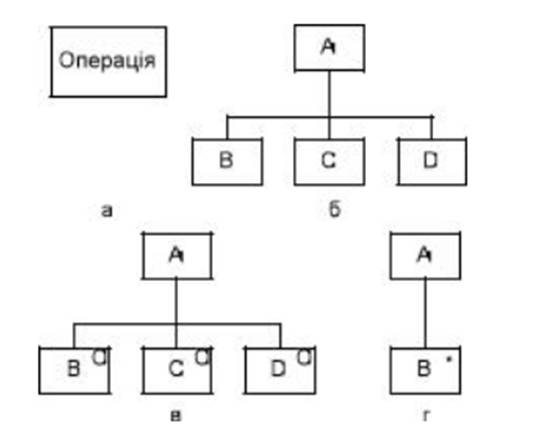
Практично, будь-який клас систем успішно моделюється за допомогою DFD (Data Flow Diagrams) - орієнтованих методів. Вони із самого початку створювалися як засіб проектування ІС, тоді як SADT - (Structure Analysis and Design Technique) як засіб моделювання систем взагалі, і мають багатший набір елементів, що адекватно відображають специфіку таких систем.

***Метод SADT*** є сукупністю правил і процедур, призначених для побудови функціональної моделі об'єкта певної предметної області. Функціональна модель SADT відображає функціональну структуру об'єкта, тобто його дії і зв'язки між цими діями. Метод SADT розроблений Дугласом Россом у 1969 р. для моделювання штучних систем середньої складності. Цей метод успішно використовувався у військових, промислових і комерційних організаціях США для вирішення широкого кола завдань, таких як довгострокове і стратегічне планування, автоматизоване виробництво і проектування, розробка ПЗ для оборонних систем, управління фінансами і матеріально-технічним постачанням тощо. Метод SADT підтримується Міністерством оборони США, яке було ініціатором розробки сімейства стандартів IDEF (Icam DEFinition), які є основною частиною програми ІСАМ (інтегрована комп'ютеризація виробництва), що проводиться за ініціативою BBC США.

Структури на діаграмах Джексона будуються з чотирьох основних компонентів, поданих на рис. 1:

* операція – блок кодів, що має один вхід і один вихід (рис. 1, а);
* проходження – послідовне виконання операцій зліва направо (рис.1,б);
* вибір – виконання однієї з операцій залежно від виконання умови (рис. 1, в);
* ітерація – багатократне виконання блоку (рис. 1, г).

**Рис 1.** Елементи структурних діаграм Джексона.

У методі потоків даних ПС розглядається як перетворювач вхідних потоків у вихідні. Метод потоків даних, як і структурний метод, з успіхом застосовувався при вирішенні ряду складних завдань, зокрема, в системах інформаційного забезпечення, де існують прямі зв'язки між вхідними та вихідними потоками системи і де не потрібно приділяти особливу увагу швидкодії.

*Методи об'єктно орієнтованого аналізу і проектування ПЗ. Мова UML.*

Концептуальною основою об'єктно орієнтованого аналізу і проектування ПЗ (ООАП) є *об'єктна модель****,*** її основні принципи - (абстрагування, інкапсуляція, модульність та ієрархія) і поняття - (об'єкт, клас, атрибут, операція, інтерфейс тощо). В основі об'єктно-орієнтованого проектування (ООП) лежить уявлення про те, що ПС необхідно проектувати як сукупність взаємодіючих між собою об'єктів, розглядаючи кожен об'єкт як екземпляр певного класу, причому класи утворюють ієрархію. Об'єктно-орієнтований підхід відображає топологію новітніх мов високого рівня, таких як Smalltalk, Object Pascal, C ++, CLOS і Ada.

Більшість сучасних методів ООАП базуються на використанні уніфікованої мови моделювання UML (Unified Modeling Language), яка є мовою для визначення, подання, проектування і документування програмних систем, організаційно-економічних систем, технічних систем та інших систем різної природи. UML містить стандартний набір діаграм і нотацій найрізноманітніших видів.

UML - це наступник того покоління методів ООАП, які з'явилися в кінці 1980-х і на початку 1990-х років. Створення UML фактично розпочалося в кінці 1994 p., коли Граді Вуч і Джеймс Рамбо почали роботу щодо об'єднання їх методів Booch і ОМТ (Object Modeling Technique) під егідою компанії Rational Software. До кінця 1995 р. вони створили першу специфікацію об'єднаного методу, названого ними Unified Method. Тоді ж у 1995 р . до них приєднався автор методу OOSE (Object-Oriented Software Engineering) Івар Якобсон. Таким чином, UML є прямим об'єднанням і уніфікацією методів Г. Буча, Д. Рамбо і Г. Якобсона, проте доповнює їх новими можливостями.

Головними при розробці UML були такі цілі:

* надати користувачам готову до використання виразну мову візуального моделювання, що дозволяє їм розробляти осмислені моделі й обмінюватися ними;
* передбачити механізми розширюваності і спеціалізації для розширення базових концепцій;
* забезпечити незалежність від конкретних мов програмування і процесів розробки;
* забезпечити формальну основу для розуміння цієї мови моделювання (мова має бути одночасно точною і доступною для розуміння, без зайвого формалізму);
* стимулювати зростання ринку об'єктно орієнтованих інструментальних засобів;
* інтегрувати кращий практичний досвід.

UML прийнятий на озброєння практично всіма найбільшими компаніями - виробниками ПЗ (Microsoft, Oracle, IBM, Hewlett-Packard, Sybase тощо). Крім того, практично всі світові виробники CASE-засобів, крім IBM Rational Software, підтримують UML у своїх продуктах (Oracle Designer, Together Control Center (Borland), AllFusion Component Modeler (Computer Associates), Microsoft Visual Modeler). Стандарт UML версії 1.1, прийнятий OMG у 1997 p., містить такий набір діаграм: '

* діаграми класів (class diagrams) - для моделювання статичної структури класів системи і зв'язків між ними;
* діаграми компонентів (component diagrams) - для моделювання ієрархії компонентів (підсистем) системи;
* послідовність виконуваних дій;
* передачу інформації між функціональними процесами;
* відношення між даними.

*Методи прототипування (Prototyping Methods)* засновані на використанні прототипу ПЗ для моделювання на ньому завдань нової системи і базуються на:

– стилях прототипування, що уособлюють тривалість використання прототипів, наприклад, стиль створення тимчасово використовуваних прототипів (throw away),

– моделях еволюційного прототипування – перетворення прототипу в кінцевий продукт і розроблення специфікацій, відповідно до якої він виконується;

– техніках оцінки/дослідження (evaluation) результатів прототипування.

*Методи прототипування* використовуються для отримання прототипів - проміжних версій ПС. Прототипи створюються в деяких процесах розробки для демонстрації концепцій, закладених в ПС, перевірки варіантів вимог, виявлення проблем, які можуть виникнути і в ході розробки, і при експлуатації системи.

Методи дозволяють відповісти на питання, як створити програмний виріб, і охоплюють круг задач, пов’язаних з плануванням розробки, оцінкою термінів і витрат, з аналізом вимог, проектуванням, кодуванням і тестуванням програм, з супроводом програмного виробу. Метод програмної інженерії – це структурний підхід до створення ПЗ, націлений на створення ефективного продукту найбільш прибутковим (рентабельним) шляхом. Практично всі методи побудовані на ідеї створення графічних моделей системи з подальшим їх використанням в якості специфікації, або архітектури системи. Методи повинні включати в себе наступні компоненти:

* опис моделей системи і їх нотацій;
* правила, які накладають обмеження на використання моделей в системі;
* рекомендації, які характеризують хороші прийоми проектування в даному методі;
* керівництво до дії – опис дій, яким можна слідувати під час створення моделей і подальшого їх використання.

Компоненти методів інженерії програмного забезпечення

| **компонент** | **зміст** | **приклад** |
| --- | --- | --- |
| Опис моделі системи | Опис моделей створюваних систем і графічна нотація, використовувана для розробки цих моделей | Моделі об'єктів, моделі статичної структури даних, моделі потоків даних, моделі зміни станів системи і т. п. |
| Правила | Правила та обмеження, які необхідно виконувати при розробці системи | Кожен елемент системи повинен мати унікальне ім'я |
| Рекомендації | Евристичні поради та рекомендації, що відображають практичний досвід застосування даного методу | Будь-який об'єкт моделі не повинен мати більше семи підлеглих йому об'єктів |
| Керівництво по застосуванню методу | Опис робіт, які необхідно виконати для побудови моделі системи, а також рекомендації щодо організації цих робіт | Атрибути будь-якого об'єкта повинні бути задокументовані, перш ніж будуть визначені операції, пов'язані цим об'єктом |

Немає ідеальних методів, усі вони застосовуються лише у тих чи інших випадках. Наприклад, об’єктно-орієнтовані методи добре підходять для інтерактивних систем, але не для систем реального часу.

Методи розробки надзвичайно важливі для подолання складності програмування системи, так як вони впорядковують процес створення складних систем, і як спільні кошти доступні для всієї команди розробників. Крім того, метод - це послідовний процес створення моделей, який описує певним чином різні сторони розробляється програмної системи. Їх застосування дозволяє менеджерам в процесі розробки оцінити ступінь просування і ризик.

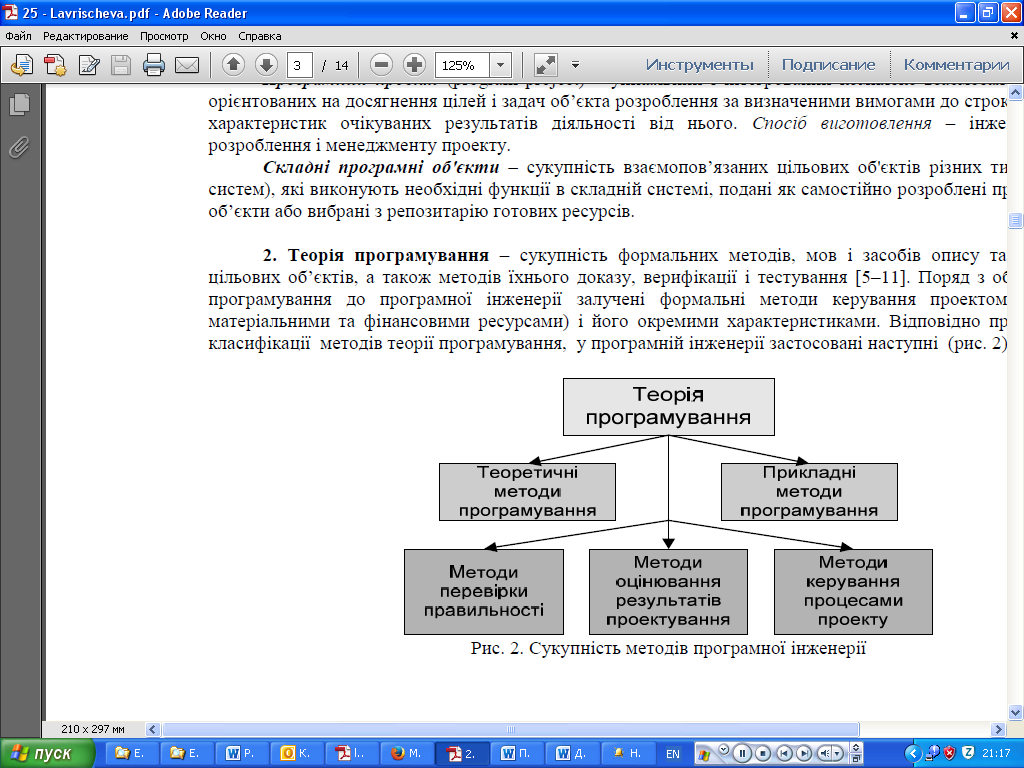
Сукупність формальних методів, мов і засобів опису та проектування цільових об’єктів, а також методів їхнього доказу, верифікації і тестування надає ***теорія програмування***. У ПІ застосовані такі методи теорії програмування:

– методи програмування ***теоретичні*** (алгоритмічний, експлікативний[[1]](#footnote-1), функціональний, алгебраїчний, VDM[[2]](#footnote-2), RAISE[[3]](#footnote-3) 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).



Поряд з методами теорії програмування до ПІ залучені формальні методи керування проектом (персоналом, матеріальними та фінансовими ресурсами) і його окремими характеристиками.

*Створені теорії об****’****єктного і компонентного програмування****,*** *спадкоємці модульного програмування, спрямовані на виробництво з готових “деталей” або ресурсів складних ПС. Теорія об’єктного, компонентного програмування та перетворення типів даних різномовних компонентів використовується в інженерії виробництва складних програм з готових компонентів повторного використання (ПВК).*

Перевірки правильності компонентів **/** 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*** – метод виявлення при виконанні вихідного коду за тестовими даними різних помилок, дефектів, відмовлень і збоїв, викликаних нерегулярними ситуаціями або аварійним припиненням роботи системи.

***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”.

***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.

**ЕМПІРИЧНІ МЕТОДИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ**

При емпіричних дослідженнях в ПІ використовують методи:

- абстрагування,

- аналіз,

- синтез,

- індукція,

- дедукція,

- моделювання.

***Абстрагування*** – це уявне відкидання неістотних, другорядних ознак предметів і явищ, зв’язків і відношень між ними, та виділення аспектів, що цікавлять дослідника. Абстракція найчастіше виділяє з явищ одну, певну сторону у чистому вигляді.

Процес абстрагування проходить дві ступені. Перша ступінь полягає у вичленовуванні найбільш важливого в явищах і встановлення незалежності або зневажено слабкої залежності досліджуваних явищ від визначених факторів (якщо А не залежить від фактора Б, то можна відвернутися від останнього як несуттєвого). Друга ступінь це реалізація можливостей абстрагування. Суть цієї реалізації полягає в тому, що здійснюється заміщення об'єкта К1 іншим, менш багатим властивостями об'єктом К2, що виступає в якості "моделі" першого.

Абстракції існують наступних основних типів:

– *ототожнення* - утворення понять шляхом об'єднання предметів, зв'язаних відносинами типу рівності в особливий клас (відволікання від ряду індивідуальних властивостей предметів);

– *ізолювання* - виділення властивостей і відносин, нерозривно зв'язаних із предметами, і позначення їх визначеними "іменами", що дає абстракціям статус самостійних предметів ("надійність", "технологічність"). Розходження між цими двома абстракціями полягає в тому, що в першому випадку ізолюється комплекс властивостей об'єкта, а в другому - єдина його властивість;

– *конструктивізація* - відволікання від хиткості, невизначеності границь реальних об'єктів, "огрубіння" дійсності.

– *абстракція актуальної нескінченності* - це одна з основних абстракцій математики і логіки. Сутність - у відволіканні від незавершеності процесу утворення нескінченної множини, від неможливості задати його повним списком всіх елементів;

– *абстракція потенційної здійсненності* - ця абстракція також знаходить найбільше застосування в математиці і логіці. Суть її у відволіканні від реальних границь людських можливостей, обумовлених обмеженістю життя в часі і просторі. Нескінченність виступає вже не як безпосередньо дане, актуальне, а як потенційно здійсненне.

***Аналіз*** - це метод пізнання, що дозволяє розчленовувати предмети дослідження на складові частини з метою його детального вивчення.

***Синтез*** є об’єднанням окремих частини в окреме ціле.

Прямий, або емпіричний аналіз і синтез, використовується при поверховому аналізі, при цьому здійснюється виділення окремих частин об’єкта, виявлення його властивостей проведення найпростіших вимірювань, які поверхово характеризують об’єкт, предмет чи елемент. Цей вид аналізу чи синтезу дає можливість ознайомитись з явищем чи синтезом але не дозволяє вникнути в сутність.

Зворотний, або теоретичний аналіз і синтез використовує для дослідження сутності явища об’єкта чи предмета. Тут оперативний аналіз чи синтез базуються на теоретичних дослідженнях, припущеннях, причинно-наслідкових зв’язках.

У наукових судженнях встановлюють зв’язки між предметами чи явищами або між їхніми певними ознаками. Шлях до дослідження проходить через безпосереднє сприйняття предмета та явища, а також їхніх зв’язків.

Існує два основних вид досягнення висновків:

− індуктивний (індукція),

− дедуктивний (дедукція).

***Індукція*** – це метод досягнення висновку, де умови ґрунтуються на перехід від часткового до загального. Тут на основі знань про частини основних класів робиться висновок про клас в цілому. Як метод дослідження індукція це процес дослідження вивчення явищ під час якого здійснюється перехід від окремих факторів до загальних положень.

***Дедукція*** – це такий метод досягнення висновку, у якому висновок про елемент множини робиться на основі значення про загальні властивості всієї множини. У випадку дедуктивного методу висновки повинні бути одержані на основі використання методів логічного аналізу. Змістом дедукції, як наукового пізнання, є застосування загальних наукових положень при дослідженні конкретних явищ. Важливою передумовою дедукції у практиці є пізнання конкретних завдань.

***Моделювання*** – це непрямий, опосередкований метод дослідження об’єктів, явищ елементів, які ґрунтуються на застосуванні моделі як засобу досліджень. Під моделлю розуміють уявну або матеріально реалізовану систему яка при відображенні чи відтворенні об’єкта здатна замістити його так, що сама стає джерелом інформації про об’єкт дослідження.

Основна функція моделі – бути засобом пізнання. Виділяють наступні похідні функції моделей:

– засіб осмислення дійсності,

– засіб спілкування,

– засіб навчання та тренажу,

– інструмент прогнозування,

– засіб постановки та проведення експериментів.

*Модель як засіб осмислення дійсності* дозволяє впорядкувати, та, при можливості, формалізувати первинні нечіткі або суперечливі уявлення про те чи інше явище, систему. У процесі побудови моделі виявляються різні взаємні залежності та відповідності. Так, наприклад, при побудові інформаційної моделі підприємства, спочатку будують так звану фактографічну модель, що дозволяє виявити непотрібні дублювання, обґрунтування зміни в оргструктурі, оптимізувати документообіг, після чого можна приступати до проектування інформаційної системи.

Як *засіб спілкування* модель дозволяє точніше описати складні поняття, порівняно з нечітким словесним описом, описує систему стисліше, дозволяє зрозуміти причинно-наслідкові зв’язки та загальну структуру системи, що моделюється.

Використання моделей *для навчання та тренажу* дозволяє підвищити ефективність та скоротити терміни навчання. Імітація різноманітних практичних ситуацій на моделі, особливо ситуацій критичних, інформація про дії в яких збагачена досвідом попередників, сприяє підвищенню якості навчання. На практиці широко використовуються різноманітні тренажери для навчання водіїв, льотчиків, космонавтів, працівників енергосистем. Дуже важливе застосування моделей – ділові ігри для навчання персоналу.

Одним з найчисленніших є використання моделей для *прогнозування*, передбачення на основі інформації про минулу поведінку системи її поведінки в майбутньому.

Як *засіб проведення експерименту* модель використовується в тих випадках, коли проведення експериментів на реальній системі недоцільне, або неможливе. Наприклад, вибір оптимальної структури системи прийняття рішень шляхом експериментування на реальному підприємстві призведе до великих витрат. Випробування літака в критичних режимах загрожує життю пілота, а тому припустимі межі необхідно оцінити за результатами експериментів на моделі.

*Класифікація моделей* здійснюється за різними класифікаційними ознаками: ступінь визначеності, область зміни параметрів та змінних моделі, фактор часу, засоби опису та оцінки, природа моделей.

*За ступенем визначеності* *моделі* класифікуються як детерміновані, стохастичні, та з невизначеністю.

Характерним для детермінованих моделей є те,що при певних значеннях вхідних змінних на виході моделі можна отримати лише один результат. Детермінована модель може відображати як детерміновану, так і стохастичну систему, в останньому випадку зі спрощеннями та абстрагуванням від випадкових факторів.

У стохастичних моделях змінні, параметри, умови функціонування та характеристики стану системи є випадковими величинамита пов’язані стохастичними (випадковими) залежностями. Тому характеристики стану та реакції в моделі визначаються законами розподілу ймовірності їх виникнення. В процесі побудови стохастичних моделей для отримання характеристик моделі та опрацювання результатів моделювання широко використовуються методи регресійного, кореляційного, та факторного аналізу.

В моделях з невизначеністю розподіл ймовірностей певних параметрів може або взагалі не існувати, або ж бути невідомим.

*За областю зміни параметрів* *моделі* можуть бути дискретні, неперервні та дискретно-неперервні.

Характерним для дискретної моделі є те, що множини припустимих тзначень змінних та параметрів в ній дискретні. Дискретна модель може відображати як дискретні, так і неперервні системи, які в цьому випадку представляються в дискретному вигляді шляхом введення різноманітного типу шкал, бальних оцінок та ін..

У неперервних моделях всі змінні та параметри моделі є неперервними, такими моделями є, наприклад, модель у вигляді системи диференціальних рівнянь.

*За фактором часу* розрізняються статичні та динамічні моделі. В статичній моделі всі залежності стосуються одного моменту часу. Прикладом статичної моделі може бути модель структури системи, як незмінної в часі характеристики. В статичних моделях відсутні залежності від часу в явному вигляді. У динамічних моделях значення змінних явно залежить від часу.

*За засобами описування та оцінювання* розрізняють дискрептивні та нормативні моделі. Дискрептивні моделі не включають наочно сформульованого критерія оцінки якості функціонування об’єкту, що моделюється. Нормативні моделі включають такі критерії.

*За природою* моделі поділяють на предметні та знакові.

Предметні моделі – це фізичні тіла або системи. Деякі моделі є природніми, інші – штучні.

Природні моделі поділяють на живі, неживі, екологічні, соціальні. Штучні предметні моделі – це натурні та аналогові моделі. Моделі, які нагадують реальну систему – макети в натуральну величину або зменшені в певному масштабі належать до натурних. Натурне моделювання може також реалізовуватися й на частинах системи. В аналогових моделях властивість реальної системи представляється деякою іншою властивістю аналогічної за поведінкою моделі.

В знакових (символічних, абстрактних) моделях для представлення моделі використовуються символи, а не фізичні пристрої. Знакові моделі підрозділяються на мовні, в яких система описується за допомогою формалізованої або напівформалізованої мови, та математичні, в яких поведінка об’єкту, що моделюється, та зв’язки між його елементами описуються засобами математики. Знакові моделі досліджує спеціальна область знань – семіотика. Семіотика вивчає знаки не окремо, а як такі, що входять в знакові системи, в яких виділено 3 групи відношень:

– синтаксис (будова) – це відношення між різними знаками, що дозволяє їх розрізняти та будувати з них складніші знакові конструкції;

– семантика (позначення) – це відношення між знаками та тим, що вони визначають;

– прагматика (дія) – відношення між знаками і тими, хто їх використовує в своїй діяльності, сенс.

Математичні моделі будуються двох типів – аналітичні та імітаційні.

Аналітичні моделі описують функціонування системи у вигляді певних функціональних залежностей та (або) логічних співвідношень. Приклади таких моделей – система алгебраїчних рівнянь, що описує міжгалузевий баланс народного господарства; Система інтегродиференційних рівнянь, яка описує процеси перерозподілу енергії в енергетичних мережах.

Імітаційні моделі відтворюють процес функціонування системи в часі шляхом моделювання елементарних явищ в системі, обміну сигналами між елементами системи, формування вихідних сигналів за зміни станів елементів. Імітаційні моделі дозволяють врахувати такі різнорідні властивості елементів системи, як неперервність та дискретність, детермінізм та стохастичність, лінійність та не лінійність. Концепція ООП, власне, реалізує парадигму імітаційного моделювання.

Серед *методів теоретичних досліджень* можна виділити наступні:

- історичний;

- системний;

- логічний;

- когнітивний.

Дані методи, застосовуються для вирішення і аналізу складних математичних задач за допомогою використання інформаційних технологій. До методів теоретичних досліджень також можна віднести:

- метод сходження від абстрактного до корректного;

- метод формалізації;

- аксіоматичний метод.

***Формалізація*** це метод вивчення різноманітних об’єктів, явищ, елементів, зв’язків шляхом їх відображення та відображення їх структури за допомогою штучних мов, в переважно математичних.

Переваги формалізації:

– забезпечує узагальненість до вирішення проблем;

– стислість, чіткість подачі інформації;

– однозначність інформації, тобто уникаємо багатозначності звичної мови.

– формалізація дає змогу формалізувати знакові моделі об’єктів і замінювати значення реальних процесів при дослідженні моделей.

Серед великої різноманітності загальних наукових методів окремо виділяють історичний і логічний які дозволяють відтворити досліджуваний об’єкт і уявити собі як він розвивався. Загально науковий статус має:

- математичний (кількісне визначення предметів та явищ),

- аксіоматичний,

- статистичний,

- системно–структурний,

- кібернетичний,

- теоретично інформативні методи досліджень.

*Аксіоматичний метод* – це засіб побудови, при якому без доведення приймаються деякі твердження аксіоми, а потім використовуються для підтвердження інших правил теорем, логічних правил. Аксіоматичний підхід є одним з найпоширеніших формальних підходів. Необхідними припущеннями при побудові аксіоматичних моделей систем є:

– достатність математичної моделі для всебічного дослідження відповідно до мети моделювання у визначених умовах;

– наявність множини базових припущень щодо найхарактерніших форм та виявів внутрішніх системних механізмів, що повинні бути вивчені, або процедури їх виявлення.

Таким чином, математична модель будується на грунті використання базових припущень (аксіом), які, в свою чергу, повинні відповідати наступним вимогам:

– базові припущення створюються, використовуючи наявне змістовне описання функціонування системи в аспектах, що підлягають пізнанню;

– базові припущення повинні утворювати абстрактну систему, до складу якої входитимуть: визначення – терміни деякої формальної мови описання змінних предметної області; вирази – складаютьсяз термінів та символів математичних операцій, які є або дискрептивним визначенням характеристик властивостей системи, або правила, що встановлюють закони формального виведення нових виразів;

– система базових припущень повинна бути формально та змістовно несуперечливою системою аксіом.

Процес побудови аксіоматичної моделі полягає у відповідній інтерпретації та переведенні змістовного описання системи мовою точних математичних термінів та відношень, в процесі чого усуваються неясності, суперечності, неповнота або надлишковість, які властиві змістовному описанню системи. Однак ця тягне за собою в деяких випадках ряд суттєвих спрощень, які можуть змінити уявлення про об’єктивні механізми функціонування реальної системи, що моделюється. Аксіоматичні моделі як абстрактні системи широко застосовуються в природничих та технічних науках для побудови моделей реального світу, а також в математиці для отримання та дослідження абстрактних об’єктів з новими властивостями. Аксіоматичний підхід добре зарекомендував себе при дослідженні систем, які діють відповідно до визначених правил у детермінованому середовищі. За допомогою цього підходу будуються абстрактні системи, що можуть бути застосовані до широкого кола проблем, які можуть бути формалізованими. Разом з тим, часто виникають складні задачі та проблеми, де безпосереднє застосування аксіоматичного підходу є неможливим. Тому частіше і з успіхом аксіоматичний підхід застосовується до аналізу та опису функціонування окремих підсистем та елементів складної системи.

*Емпірико-статистичні моделі* та методи широко використовують кібернетичну ідею “чорної скрині”, що відкидає інформацію про структуру системи. Сама назва образно підкреслює повну відсутність інформації про внутрішню будову: в цій моделі задані, фіксуються та перераховуються лише вхідні та вихідні зв’язки з середовищем. У багатьох випадків достатнім є змістовне описання входів та виходів – в цьому випадку “чорна скриня” – це список входів та виходів. При детальнішому підході може виявитись необхідність кількісного описання характеристик входів та виходів. Цей підхід також виявляється зручним при проектуванні систем, в тому числі програм – на деякому етапі треба добре зрозуміти що є входами та виходами. А вже потім приймаються рішення про способи реалізації необхідного функціоналу.

***Засоби і інструменти ПІ*.**

Методи програмної інженерії - це структурні рішення, призначені для розробки програмного забезпечення і включають системні моделі, формалізовані нотації і правила проектування, а також способи управління процесом розробки.

В якості основи для реалізації методів програмної інженерії, як правило, використовуються CASE-засоби. Сукупність методів, застосовуваних в життєвому циклі розробки програмного забезпечення та об'єднаних одним загальним філософським підходом, являє собою методологію розробки.

***Засоби (утиліти) технології конструювання ПЗ*** забезпечують автоматизовану або автоматичну підтримку методів. Сьогодні існують засоби практично для кожного з методів. Коли ці засоби об’єднуються в інтегроване середовище так, що інформація, отримана одним з них, може використовуватися іншим, створюється система підтримки розробки програмного забезпечення. Такі системи називають CASE системами (CASE – Computer Aided Software Engineering – програмна інженерія з комп’ютерною підтримкою).

Поняття CASE включає широкий комплекс програм, призначених для підтримки процесів створення програмного продукту, включаючи аналіз вимог, моделювання, наладку і тестування. Більшість сучасних методів підтримується відповідними CASE-засобами, які поділяються на дві категорії: перші, які підтримують аналіз і проектування – CASE-засоби верхнього рівня, і інші, які підтримують реалізацію і тестування, такі як наладчики, засоби аналізу системи, генератори тестів і редактори програм – CASE-засобами нижнього рівня.

Процедури об’єднують в єдиний технологічний процес методи і засоби так, що дозволяють найбільш раціонально і ефективно вести розробку програмної продукції. Процедури встановлюють послідовність використання методів, специфіку представлення необхідних документів, форм і звітів, послідовність виконання контрольних і управляючих дій.

CASE-засоби - це програмне забезпечення, що підтримує процеси життєвого циклу ПЗ, у тому числі аналіз вимог до системи, проектування прикладного ПЗ і БД, генерацію коду, тестування, документування, забезпечення якості, управління конфігурацією ПЗ і проектом, а також інші процеси (відповідно до міжнародного стандарту ISO/IEC 14102:1995(Е)). CASE-засоби разом із системним ПЗ і технічними засобами утворюють середовище розроблення ПЗ.

Найбільш трудомісткими стадіями розроблення ПЗ є стадії формування вимог і проектування, у процесі яких CASE-засоби забезпечують якість прийнятих технічних рішень і підготовку проектної документації. При цьому велику роль відіграють методи візуального подання інформації. Це передбачає побудову різноманітних графічних моделей, наскрізну перевірку синтаксичних правил тощо. Графічні засоби моделювання ПЗ дають змогу розробникам наочно вивчати функціонуючу ІС, перебудовувати її відповідно до поставлених цілей і обмежень.

Сучасний ринок програмних засобів нараховує близько 300 САSЕ-засобів. САSЕ-засобам властиві такі основні особливості:

* наявність потужних графічних засобів для опису і документування системи, які забезпечують зручний інтерфейс з розробником;
* інтеграція окремих компонентів САSЕ-засобів для забезпечення управління процесом розроблення ПЗ;
* використання організованого сховища проектних мета-даних.

Інтегровані САSЕ-засоби, що підтримують повний ЖЦ ПЗ, містять такі компоненти:

* репозиторій, який має забезпечувати збереження версій проекту та його окремих компонентів, синхронізацію надходження інформації від різних розробників при груповій розробці, контроль метаданих за повнотою і несуперечністю;
* графічні засоби аналізу і проектування, що забезпечують створення й редагування комплексу діаграм, які утворюють моделі діяльності організації і системи ПЗ;
* засоби розроблення застосувань, включаючи мови 4GL і генератори кодів;
* засоби управління вимогами;
* засоби управління конфігурацією ПЗ;
* засоби документування;
* засоби тестування;
* засоби управління проектом;
* засоби реверсного інжинірингу ПЗ і БД.

Основні функції засобів організації і підтримки репозиторію - збереження, доступ, відновлення, аналіз і візуалізація всієї інформації з проекту ПЗ. Репозиторій містить не тільки інформаційні об'єкти різних типів, але і зв'язок між їх компонентами, а також правила використання та обробки цих компонентів. Репозиторій може зберігати понад 100 типів об'єктів, прикладами яких є діаграми, визначення екранів і меню, проекти звітів, опису даних, вихідні коди тощо.

Інформаційні об'єкти репозиторію описуються перерахуванням їх властивостей: ідентифікатор, імена-синоніми, тип, текстовий опис, компоненти, область значення. Крім цього, зберігаються усі зв'язки з іншими об'єктами, правила формування і редагування об'єкта, а також контрольна інформація про час створення об'єкта, час його останнього відновлення, номери версії, можливості відновлення тощо.

Репозиторій є базою для стандартизації проектної документації і контролю проектних специфікацій. Важливі функції управління й контролю проекту також реалізуються на основі репозиторію. Зокрема, за допомогою репозиторію може здійснюватися контроль безпеки (обмеження доступу, привілеї доступу), контроль версій, контроль змін.

Графічні засоби забезпечують:

* створення ієрархічно пов'язаних діаграм, у яких поєднані графічні і текстові об'єкти;
* створення і редагування об'єктів у будь-якому місці діаграми;
* створення, переміщення і вирівнювання груп об'єктів, зміну їхніх розмірів, масштабування;
* збереження зв'язків між об'єктами при їх переміщенні і зміні розмірів;
* автоматичний контроль помилок.

Важливість контролю помилок на стадіях формування вимог і проектування зумовлена тим, що на більш пізніх стадіях їхнє виявлення й усунення обходяться значно дорожче. У САSЕ-засобах звичайно реалізуються такі види контролю:

* контроль синтаксису діаграм і типів їхніх елементів;
* контроль повноти діаграм: усі елементи діаграм мають бути ідентифіковані і відображені в репозиторії;
* наскрізний контроль діаграм одного або різних типів щодо їхньої здійснюваності за рівнями - ***вертикальне і горизонтальне балансування діаграм.***

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 (С++, Java, Pasсal 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 (Rational Unified Process), MSF (**Microsoft Solution Framework)**, 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.

**Інструменти інженерії ПЗ** забезпечують автоматизовану підтримку процесів розроблення ПЗ і містять у собі множину інструментів, що охоплюють усі процеси ЖЦ.

*Інструменти роботи з вимогами (Software Requirements Tools)* – це:

– інструменти розробки (Requirement Development) і керування вимогами (Requirement Management), орієнтовані на аналіз, збирання, специфікування і перевірку вимог;

– інструменти трасування вимог (Requirement traceability tools) є невід'ємною частиною роботи з вимогами, їх функціональний зміст залежить від складності проектів і рівня зрілості процесів.

*Інструменти проектування (Software Design Tools) –* це інструменти для створення ПЗ із застосуванням базових нотацій (структурної SADT/IDEF, моделювання UML і т.п.).

*Інструменти конструювання ПЗ (Software Construction Tools) –* це інструменти для трансляції і об’єднання програм. До них належать:

– редактори програм (program editors) і програми редагування загального призначення;

– компілятори і генератори коду (compilers and code generators) як самостійні засоби об'єднання програмних компонентів в інтегрованому середовищі для одержання вихідного продукту з використанням препроцесорів, складальників, завантажників і ін.;

– інтерпретатори (interpreters), які забезпечують контрольоване виконання програм за їх описом. Намітилася тенденція злиття інтерпретаторів і компіляторів (наприклад, Java, в .NET);

– відлагоджувачі (debuggers), призначені для перевірки правильності опису вихідних програм і усунення помилок;

– інтегроване середовище розробки (IDE – integrated development environment) та бібліотеки компонентів (libraries components), що є утворюють середовище виконання процесу розроблення ПС;

– програмні платформи (Java, J2EE і Microsoft .NET) і платформи для розподілених обчислень (CORBA і WebServices, тощо).

*Інструменти тестування (Software Testing Tools) –* це:

– генератори тестів (test generators), що допомагають у розробці сценаріїв тестування;

– засоби виконання тестів (test execution frameworks), які забезпечуютьвиконання тестових сценаріїв і відслідковують поведінку об'єктів тестування;

– інструменти оцінки тестів (test evaluation tools), які підтримують оцінювання результатів виконання тестів і ступеня відповідності поведінки тестованого об'єкта очікуваній поведінки;

– засоби керування тестами (test management tools), які забезпечують інженерне керування процесом тестування ПЗ;

– інструменти аналізу продуктивності (performance analysis tools), кількісної її оцінки та оцінки поводження програм у процесі виконання.

*Інструменти супроводу (Software Maintenance Tools)* містять у собі:

– інструменти полегшення розуміння (comprehension tools) програм, наприклад, різні засоби візуалізації;

– інструменти реінженерії (reengineering tools) підтримують діяльність з перетворення програм і зворотної інженерії (reverse engineering) для відновлення (артефактів, специфікації, архітектури) застарілого ПЗ або генерації нового продукту.

*Інструменти конфігураційного керування (Software Configuration Management Tools)* – це:

– інструменти відстеження (tracking) дефектів;

– інструменти керування версіями;

– інструменти керування складанням, випуском версії (конфігурації) продукту та його інсталяції.

*Інструменти керування інженерною діяльністю (Software Engineering Management Tools)* підрозділяються на:

– інструменти планування і відстеження ходу проектів, кількісної оцінки зусиль і вартості робіт у проекті (наприклад, Microsoft Project 2003);

– інструменти керування ризиками, які використовуються для ідентифікації , моніторингу ризиків і оцінки нанесеного ушкодження;

– інструменти кількісної оцінки властивостей ПЗ шляхом вимірювань і розрахунків остаточного значення надійності і якості.

*Інструменти підтримки процесів (Software Engineering Process Tools)* розділені на:

– інструменти моделювання та опису моделей ПЗ (наприклад, UML і його інструменти);

– інструменти керування програмними проектами (наприклад, Microsoft Project);

– інструменти керування конфігурацією для підтримки версій і всіх артефактів проекту.

*Інструменти забезпечення якості (Software Quality Tools)* діляться на дві категорій:

– інструменти інспектування для підтримки перегляду (review) і аудиту;

– інструменти статичного аналізу артефактів, даних, потоків робіт і перевірки їх властивостей на відповідність показникам.

*Додаткові аспекти інструментального забезпечення (Miscellaneous Tool Issues)* стосуються:

– техніки інтеграції інструментів (платформ, представлень, процесів, даних) для їх природного сполучення в інтегрованому середовищі;

– метаінструментів для генерації інших інструментів для ПЗ;

– оцінки інструментів при їх еволюції.

*Для самостійного вивчення*: класифікація в swebok загальних інструментів, рекомендованих для застосування стосовно всіх видів об’єктів у процесах ЖЦ. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

*Контрольні запитання*.

1. Визначте основні цільових об’єктів ПІ та їх спосіб виготовлення.
2. На які категорії поділяють методи ПІ?
3. В яких галузях рентабельні формальні методи?
4. Які методології евристичних методів Вам знайомі?
5. Коли доцільно використовувати методи прототипування?
6. Визначте компоненти методів інженерії ПЗ?
7. Які методи теорії програмування використовує ПІ?
8. Які методи та інструменти використовуються для перевірки правильності компонентів **/** програм?
9. Як визначається масштаб проекту?
10. Які методи використовують в емпіричних дослідженнях ПІ ?
11. Поясніть призначення CASE систем?
12. На які групи можна поділити інструменти інженерії ПЗ?

*Література*.

1. Бабенко Л.П., Лавріщева К.М*.* Основи програмної інженерії.– Навч. посібник.–К.: Знання, 2001.– 269с.
2. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
3. Проектування інформаційних систем: Посібник // За редакцією Пономаренка В.С. – К.: Видавничий центр "Академія". 2002. ­ 488 с. URL: <http://www.dut.edu.ua/uploads/l_874_10304054.pdf>.
4. Соммервиль И. Инженерия программного обеспечения, 6 изд. – И.д. "Вильямс", 2002. URL: <https://www.studmed.ru/download/sommervill-ian-inzheneriya-programmnogo-obespecheniya_4935164f089.html>
5. Боэм Б. Инженерное проектирование программного обеспечения. — М.: Радио и связь, 1985.
6. Вендров A.M. CASE-технологии. Современные методы и средства проектирования информационных систем. - М.: Финансы и статистика, 1998.

1. ***Експлікативне програмування***  інтегрує в собі всі найбільш суттєві парадигми (стилі) програмування (структурне, функціональне, об'єктно-орієнтоване і ін.) в рамках концептуально єдиної експлікативної платформи, основу якої становлять три основні типи об'єктів: власне об'єкти, засоби побудови з одних об'єктів інших (функції) і програмологічні засоби застосування методів побудови (композиції). [↑](#footnote-ref-1)
2. **Ві́денський ме́тод розро́бки** (англ. *Vienna Development Method*, *VDM*) — набір технологій для моделювання комп'ютерних систем, аналізу створених моделей і переходу до деталізованого проектування та програмування. [↑](#footnote-ref-2)
3. **RAISE** (англ. Rigorous Approach to Industrial Software Engineering ) включає формулювання абстрактних специфікацій, розробку їх для подальшого створення послідовних конкретних специфікацій, обґрунтування правильності розробки та переклад остаточної специфікації на мову програмування. Він заснований на ряді принципів, таких як роздільна розробка, покрокова розробка, компонування, верифікація та точність. RAISE - це абревіатура для "жорсткого підходу до інженерної індустрії програмного забезпечення", і вона дає свою назву формальній мові специфікацій, мові специфікації RAISE, пов'язаному методу та набору інструментів. [↑](#footnote-ref-3)