**Тема 1. Базові поняття, види програмного забезпечення**

**Технологія програмування** – дисципліна, що вивчає технологічні процеси програмування та порядок їх проходження.

**Системна інженерія** (System engineering) – розділ науки, що вивчає питання розробки комп'ютерних систем (архітектура, дизайн, інтеграція ПЗ та ін.).

**Програмна інженерія** (Software engineering) – дисципліна, спрямована на розробку і супровід програмного забезпечення систем, що функціонують надійно і ефективно, можуть удосконалюватися і еволюціонувати і відповідають вимогам, визначених замовником.

**Програмування** (Programming) – процес підготовки завдань для їх вирішення за допомогою комп'ютера; ітераційний процес складання програм.

**Програма** – дані, призначені для управління конкретними компонентами системи обробки інформації з метою реалізації певного алгоритму, послідовність машинних команд, призначених для досягнення конкретного результату.

**Програмне забезпечення** (ПЗ / Software) – комп'ютерні програми, процедури, а також документація і дані, що стосуються функціонування комп'ютерної системи.

Вперше термін software ввів відомий статистик Джон Тьюк (John Tukey) в 1958 році для позначення різниці апаратного забезпечення ЕОМ (hardware) від засобів обробки даних. Бьярне Страуструп (Bjarne Stroustrup) відзначив, що добре ПЗ неможливо побачити, але можна відчути, коли воно працює з помилками.

За видами виконуваних функцій програмне забезпечення підрозділяється на системне, прикладне та інструментальне. Такий поділ є умовним, оскільки широке впровадження комп'ютеризації призвело до того, що майже кожна програма має ознаки декількох видів ПЗ.

*Інтегроване середовище розробки програмного забезпечення* (Integrated development environment, IDE) - це система програмних засобів, що використовується програмістами для розробки програмного забезпечення. Як правило, середовище розробки включає текстовий редактор, компілятор і/або інтерпретатор, засоби автоматизації збирання, налагодження та різноманітні інструменти для конструювання графічного інтерфейсу користувача. Значного поширення об'єктно-орієнтованого програмування (ООП) призвело до того, що сучасні інструменти розробки включають браузер класів та інспектор об'єктів.

На сьогодні в середовища розробки підключають систему керування версіями, засоби тестування та ін.

 *Види програмного забезпечення.*

*Системне ПЗ* (System software) призначене для управління роботою комп'ютера, розподілу його ресурсів, підтримки діалогу з користувачами, а також для часткової автоматизації розроблення нових програм. Як правило, системні програми забезпечують взаємодію інших програм з апаратними складовими, організацію інтерфейсу користувача. Віділяють три типи системного ПЗ:

− операційна система (ОС) – програмне забезпечення, що забезпечує інфраструктуру, на якій можуть працювати прикладні програми. Найпоширеніші ОС – Microsoft Windows, Mac OS X та Linux;

− системи програмування – призначені для полегшення та часткової автоматизації процесу розроблення та відлагодження програм;

− сервісні програми (утиліти) – розширюють можливості ОС. До утиліт відносять архіватори, антивіруси, драйвери та ін.

*Прикладне ПЗ* (application, application software) – комп'ютерна програма, що вирішує конкретні задачі фахової діяльності користувача.

*Інструментальне ПЗ* призначене для розроблення всіх видів інформаційно-програмного забезпечення. При цьому під інформаційним забезпеченням розуміють сукупність.

Інструментальне ПЗ призначене для розробки всіх видів інформаційно-програмного забезпечення. При цьому під інформаційним забезпеченням розуміють сукупність попередньо підготовлених даних, що необхідні для роботи програмного забезпечення. До інструментального ПЗ відносять текстові редактори, системи управління базами даних, транслятори мов програмування.

*Програмне забезпечення як виріб.*

Сучасні програми вирішують найрізноманітніші завдання за змістом і галузевим значенням.

У багатьох випадках програми створюються в єдиному екземплярі для вирішення приватних дослідницьких завдань, для обробки експериментального матеріалу, прискорення обчислень, моделювання процесів, і т.д. Такі програми не мають масового застосування та доступні для використання тільки тим, хто їх розробив. Вони стають об'єктами науково-технічної творчості і рідко стають промисловими виробами.

Зовсім іншим класом програм є індустріальні програмні засоби, які можна кваліфікувати як продукцію виробничо-технічного призначення.

Вони являють собою програми на носіях даних з технічною (експлуатаційною та технологічною) документацією, розроблені відповідно до діючих стандартів і пройшли державні, міжвідомчі або відомчі випробування.

Програмні засоби, прийняті у виробництво, виготовляються за затвердженою в установленому порядку технологією. Вони повинні відповідати затвердженим технічним умовам і діючій нормативно-технічній документації, забезпечуватися гарантіями постачальника.

У цьому випадку мова йде про програмні вироби (про програмний продукт для ЕОМ).

Під програмним виробом (ПВ) розуміється універсальне програмне забезпечення, яке призначається для широкого кола користувачів, можливо, навіть не відомих заздалегідь, і повинно рекламуватися, підтримуватися в працездатному стані, розширюватися протягом тривалого періоду часу.

*Програмний виріб* – це власне програми плюс документація, гарантія якості, рекламні матеріали, навчання, поширення та супровід. Окрема машинна програма або сукупність програм і програмний виріб - далеко не одне і те ж.

Програмний виріб є продукт ретельного планування і цілеспрямованої розробки, що супроводжується чіткою документацією, який пройшов всі необхідні випробування; який описано у відповідних технічних публікаціях, розмножено у необхідній кількості примірників, обслуговується і контролюється постачальником за заздалегідь продуманим планом і може розглядатися як товар.

З питань розробки систем програмного забезпечення (але не програмних виробів) можна зробити такі припущення:

− розробник створює програмне забезпечення для себе або, принаймні, організаційно пов'язаний з користувачами програмного забезпеченні, що розробляється;

− користувач формулює свої вимоги безпосередньо розробнику, якщо останній сам не є одночасно користувачем;

− користувач бере активну участь у розробці або в обслуговуванні програмного забезпечення;

− програмне забезпечення повинно працювати тільки на певній конфігурації комплексу технічних і програмних засобів в обмеженому діапазоні змін його складу і структури даних;

− розробник сам вводить в дію програмне забезпечення у користувача;

− проблеми, що виникли при використанні програмного забезпечення, вирішуються користувачем спільно з розробником або з персоналом, що здійснює його технічне обслуговування (супровід);

− програми не мають масового застосування та доступні для використання тільки тим, хто їх розробив;

− використання програми припиняється після отримання результату.

При розробці програмного виробу (за винятком особливого випадку розробки програмного забезпечення за контрактом для єдиного користувача) можна зробити такі припущення:

− розробник не знайомий з користувачем;

− вимоги користувача формуються або розробником, або передаються йому посередницькою організацією (наприклад, поставляє програмне забезпечення);

− користувачі не беруть участь у розгляді та узгодженні проектних рішень, якщо не вважати рідкісних випадків, коли їх інтереси представлені посередниками;

− програмне забезпечення повинно зберігати працездатність в широкому діапазоні конфігурацій обчислювальних комплексів і при самих різних системних програмних засобах;

− користувачі вводять програмне забезпечення в дію або самі, або з сторонньою допомогою, але ця допомога виходить не від розробника;

− проблеми, що виникли при використанні програмного забезпечення, вирішуються шляхом листування, а іноді через посередника;

− програмний виріб призначений для широкого кола користувачів, можливо, невідомих;

− програмне забезпечення використовується багаторазово і тривалий час.

 *Технологія розробки програмного забезпечення*

 *Технології* (з грецької: ремесло + наука) – сукупність знань про способи і засоби проведення виробничих процесів.

 В одному крайньому випадку одна людина здійснює поетапну розробку програми зі свого термінала в невимушеній обстановці. Природно, він створює порівняно невеликі програми, що не вимагають особливої оцінки.

 В іншому звичайному випадку розробляється дуже складне програмне забезпечення, що призначене для функціонування в реальному масштабі часу і вимагає трудовитрат обсягами в тисячі людино-годин.

 Ці дві взаємно протилежні ситуації характеризуються різним ступенем формалізації та проведенні процесу розробки програмних засобів.

 Ступінь формалізації та проведення процесу розробки програмного забезпечення безпосередньо залежить від цілей його створення, його величини, чисельності групи розробників та інших факторів. Від того, наскільки правильно і вдало з погляду технології розробки програмного забезпечення побудовано додаток, залежить якість і життєздатність кінцевого продукту.

 Під *технологією розробки програмного забезпечення* (ТРПЗ) розуміється сукупність узагальнених і систематизованих знань, або наука про оптимальні способи (прийоми) проведення процесу розробки програмного забезпечення, що забезпечує в заданих умовах отримання програмної продукції із заданими властивостями.

 Технологія розробки програмного забезпечення являє собою інженерний підхід до розробки програмних засобів ЕОМ, що охоплює методологію програмування, проблеми забезпечення надійності програм, оцінки робочих характеристик і якості проектів.

 Технологія розробки програмного забезпечення розглядає питання управління проектуванням систем програмного забезпечення, а також засоби і стандарти розробки програм.

 Технологія розробки програмного забезпечення визначає деяку професійну культуру роботи фахівців (не тільки програмістів), що забезпечує заданий рівень продуктивності праці і якості одержуваної в результаті програмної продукції.

 Технологія розробки програмного забезпечення охоплює процес розробки програмного забезпечення від появи потреби в ньому до його виготовлення, передачі користувачеві, модифікації в процесі експлуатації і припинення його використання внаслідок морального старіння.

 В ідеалі *технологія розробки програмного забезпечення повинна задовольняти основним нижче перерахованим вимогам.*

1. Необхідна стандартизація мов проектування програм, оформлення та випробування програмних модулів, а також гарантії їх якості. Це дозволить значно скоротити дублюючі розробки, впровадити складальне програмування і вести накопичення на підприємствах і в країні високоякісного програмного продукту для його багаторазового використання як типових комплектуючих виробів.
2. Вести постійний контроль і забезпечення якості програм.
3. Програми не повинні містити неперевірених шляхів і ситуацій функціонування, які призводять до несподіваних результатів.
4. Користувачеві або покупцю програм необхідно дати чітке уявлення про можливості даної програми і технологічних умовах експлуатації, при яких гарантуються певні функції і якості.
5. Технологія розробки програмного забезпечення повинна забезпечувати відторгнення програмного виробу від його розробника, тобто людський фактор у програмуванні має бути зведений до мінімуму.
6. Технологія розробки програмного забезпечення та засоби її підтримки (автоматизації) повинні забезпечувати цілеспрямовану роботу, насамперед колективу програмістів, а не окремих особистостей. Вона повинна спонукати колектив працювати тільки правильно і злагоджено; повинна автоматично блокувати будь-які не санкціоновані технологією дії.
7. Необхідно вести акуратне документування всіх етапів розробки. Документація повинна також заноситися і зберігатися на магнітних носіях. Доступ до цієї інформації має бути відкритим, простим і автоматизованим.
8. Робота користувача повинна забезпечуватися розвиненою інформаційно- довідковою системою.
9. Засоби автоматизації технології повинні охоплювати всі етапи роботи колективу програмістів.
10. Технологія розробки програмного забезпечення повинна бути простою в освоєнні, із засобами підказки, що автоматично включаються.
11. Технологія розробки програмного забезпечення повинна мати засоби автоматичної фіксації в хронологічному порядку всіх дій, які виконуються в процесі колективного виготовлення програмного виробу - повинні вестися і зберігатися в системі журнали (протоколи, щоденники) розробки. Ці засоби повинні дозволяти відновлювати будь-який стан процесу розробки на будь-якому інтервалі виготовлення програмного продукту.

**Проблеми розробки складних програмних систем.**

**Блочно-ієрархічний підхід до створення складних систем**

**Проблеми розробки складних програмних систем.**

Більшість сучасних програмних систем об'єктивно дуже складні. Ця складність обумовлюється багатьма причинами, головною з яких є *логічна складність розв'язуваних ними завдань.*

Поки обчислювальних установок було мало і їхні можливості були обмежені, ЕОМ застосовували в дуже вузьких галузях науки і техніки, причому, в першу чергу, там, де завдання, що вирішувались, були добре детерміновані і вимагали значних обчислень. У наш час, коли створено потужні комп'ютерні мережі, з'явилася можливість перекласти на них рішення складних ресурсномістких завдань, про комп'ютеризацію яких раніше ніхто і не думав. Зараз до процесу комп'ютеризації залучаються зовсім нові предметні області, а для вже освоєних областей ускладнюються вже сформовані постановки завдань.

Додатковими факторами, що збільшують складність розробки програмних систем, є:

− складність формального визначення вимог до програмних систем;

− відсутність задовільних засобів опису поведінки дискретних систем із великою кількістю станів при недетермінованій послідовності вхідних впливів;

− колективна розробка;

− необхідність збільшення ступеня повторюваності кодів.

*Складність визначення вимог до програмних систем.*

Складність визначення вимог до програмних систем обумовлюється двома факторами. По-перше, при визначенні вимог необхідно врахувати велику кількість різних факторів. По-друге, розробники програмних систем не є фахівцями в автоматизуються предметних областях, що автоматизуються, а фахівці в предметній області, як правило, не можуть сформулювати проблему в потрібному ракурсі.

*Відсутність задовільних засобів формального опису поведінки дискретних систем.*

У процесі створення програмних систем використовують мови порівняно низького рівня. Це призводить до ранньої деталізації операцій в процесі створення програмного забезпечення і збільшує обсяг описів продуктів, що розробляються, який, як правило, перевищує сотні тисяч операторів мови програмування. Засобів же, що дозволяють детально описувати *поведінку* складних дискретних систем на більш високому рівні, ніж універсальний мова програмування, не існує.

*Колективна розробка.*

Завдяки великим обсягам проектів розробка програмного забезпечення здійснюється колективом фахівців. Працюючи в колективі, окремі фахівці повинні взаємодіяти один з одним, забезпечуючи цілісність проекту, що за відсутності задовільних засобів опису поведінки складних систем досить складно. Причому, чим більший колектив розробників, тим складніше організувати процес роботи.

*Необхідність збільшення ступеня повторюваності кодів*.

На складність програмного продукту, що розробляється, впливає і те, що для збільшення продуктивності праці компанії прагнуть до створення бібліотек компонентів, які можна було б використовувати в подальших розробках. Однак у цьому випадку компоненти приходиться робити більш універсальними, що зрештою збільшує складність розробки.

Разом узяті, ці фактори суттєво збільшують складність процесу розробки. Однак очевидно, що всі вони безпосередньо пов'язані зі складністю об'єкта розробки - програмної системи.

**Блочно-ієрархічний підхід до створення складних систем**

Практика показує, що переважна більшість складних систем як у природі, так і в техніці має ієрархічну внутрішню структуру. Це пов'язано з тим, що зазвичай зв'язки елементів складних систем різні як за типом, так і за силою, що і дозволяє розглядати ці системи як деяку *сукупність взаємозалежних підсистем*. Внутрішні зв'язки елементів таких підсистем сильніші, ніж зв'язки між підсистемами.

Наприклад, комп'ютер складається з процесора, пам'яті і зовнішніх пристроїв, а Сонячна система включає Сонце і планети, що обертаються навколо нього. У свою чергу, використовуючи ті ж розходження зв'язків, можна кожну підсистему розділити на підсистеми і т. д. до самого нижнього «елементарного» рівня, причому вибір рівня, компоненти якого слід вважати елементарними, залишається за дослідником. На елементарному рівні система, як правило, складається з небагатьох типів підсистем, по-різному скомбінованих і організованих. *Ієрархії такого типу отримали назву «ціле- частина».*

 Поведінка системи в цілому зазвичай виявляється складніше поведінки окремих частин, причому завдяки більш сильним внутрішнім зв'язкам особливості системи в основному обумовлені відносинами між її частинами, а не частинами як такими.

У природі існує *ще один вид ієрархії - ієрархія «просте-складне» або ієрархія розвитку* (ускладнення) систем в процесі еволюції. У цій ієрархії будь-яка функціонуюча система є результатом розвитку більш простої системи. Саме даний вид ієрархії реалізується механізмом успадкування об'єктно-орієнтованого програмування.

Будучи в значній мірі відображенням природних і технічних систем, програмні системи зазвичай є ієрархічними, тобто мають описані вище властивості. На цих властивостях ієрархічних систем будується *блочно- ієрархічний підхід до їх дослідження або створення.* Цей підхід передбачає спочатку створювати частини таких об'єктів (блоки, модулі), а потім збирати з них сам об'єкт.

Процес розбиття складного об'єкта на порівняно незалежні частини отримав назву *декомпозиції* .

При декомпозиції враховують, що зв'язки між окремими частинами повинні бути слабшим, ніж зв'язки елементів усередині частин. Крім того, щоб з отриманих частин можна було зібрати об'єкт, що розробляється, в процесі декомпозиції необхідно визначити всі види зв'язків частин між собою.

При створенні дуже складних об'єктів процес декомпозиції виконується багаторазово: кожен блок, у свою чергу, декомпозується на частини, поки не отримують блоки, які порівняно легко розробити.

Даний метод розробки отримав назву *покрокової деталізації.*

Істотно і те, що в процесі декомпозиції намагаються виділити аналогічні блоки, які можна було б розробляти на загальній основі. Таким чином, як уже згадувалося вище, забезпечують збільшення ступеня повторюваності кодів і, відповідно, зниження вартості розробки.

Результат декомпозиції зазвичай представляють у вигляді *схеми ієрархії*, на нижньому рівні якої розміщають відносно прості блоки, а на верхньому - об'єкт, що підлягає розробці.

На кожному ієрархічному рівні опис блоків виконують з певним ступенем деталізації, *абстрагуючись* від несуттєвих деталей. Отже, для кожного рівня використовують свої форми документації і свої моделі, що відображають сутність процесів, які виконуються кожним блоком. Так, для об'єкта в цілому, як правило, вдається сформулювати лише найзагальніші вимоги, а блоки нижнього рівня повинні бути специфіковані так, щоб з них дійсно можна було зібрати працюючий об'єкт. Іншими словами, *чим більше блок, тим більш абстрактним має бути його опис* (рис. 2.1).

Рисунок 2.1. – Співвідношення абстракт око та конкретного при описі блоків у блочно-ієрархічному підході

При дотриманні цього принципу розробник зберігає можливість осмислення проекту і, отже, може приймати найбільш правильні рішення на кожному етапі, що називають *локальною оптимізацією* (на відміну від глобальної оптимізації характеристик об'єктів, яка для дійсно складних об'єктів не завжди можлива).

*Примітка.* Слід мати на увазі, що поняття складного об'єкта в міру вдосконалення технологій змінюється, і те, що було складним вчора, не обов'язково залишиться складним завтра.

Отже, *в основі блочно-ієрархічного підходу лежать декомпозиція і ієрархічне впорядкування.* Важливу роль відіграють також такі принципи:

− несуперечність – контроль узгодженості елементів між собою;

− повнота – контроль на присутність зайвих елементів;

− формалізація – строгість методичного підходу;

− повторюваність – необхідність виділення однакових блоків для здешевлення і прискорення розробки;

− локальна оптимізація – оптимізація в межах рівня ієрархії.

Сукупність мов моделей, постановок завдань, методів описів деякого ієрархічного рівня прийнято називати *рівнем проектування.*

Кожен об'єкт у процесі проектування, як правило, доводиться розглядати з кількох сторін. Різні погляди на об'єкт проектування прийнято називати *аспектами проектування.*

Крім того, що використання блочно-ієрархічного підходу дає можливість створення складних систем, він також:

− спрощує перевірку працездатності як системи в цілому, так і окремих блоків;

− забезпечує можливість модернізації систем, наприклад, заміни ненадійних блоків із збереженням їх інтерфейсів.

Необхідно відмітити, що використання блочно-ієрархічного підходу стосовно програмним системам стало можливим тільки після конкретизації загальних положень підходу та внесення деяких змін у процес проектування. При цьому *структурний підхід* враховує тільки властивості ієрархії «ціле- частина», а об'єктний - використовує ще й властивості ієрархії «просте- складне».

Ключові терміни

*Технологія -* комплекс організаційних заходів, операцій і прийомів, які спрямовані на виготовлення, обслуговування, ремонт та / або експлуатацію виробів з номінальною якістю і оптимальними витратами.

*Технологія розробки програмного забезпечення* ***(****ПЗ****) -*** комплекс організаційних заходів, операцій і прийомів, спрямованих на розробку програмних продуктів високої якості в рамках відведеного бюджету і в строк.

*Життєвий цикл програмного забезпечення* ***(****ЖЦПЗ****) -*** період часу, який починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації.

*Каскадна* ***(****водоспадна****)*** *модель ЖЦПЗ -* послідовне виконання етапів створення ПЗ.

*Ітераційна спіральна модель ЖЦПЗ -* розробка ПЗ здійснюється по спіралі, кожен виток (ітерація) якої передбачає реалізацію певного функціоналу програмної системи.

*Інкрементна ітераційна модель ЖЦПЗ -* розробка ПЗ реалізується декількома ітераціями з поступовим нарощуванням функціональності системи.