ЛЕКЦІЯ 9. МЕНЕДЖМЕНТ КОНФІГУРАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

План лекції:

1. Основні положення менеджменту конфігурації при реалізації програмних проектів.
2. Контроль змін як одна із складових менеджменту конфігурацій.
3. Контроль версій як одна із складових менеджменту конфігурацій.
4. Контроль випусків як одна із складових менеджменту конфігурацій.
5. Поняття кортежу програми, кортежу проекту та паспорту проекту при виконанні програмних проектів.
6. Інструментальні засоби менеджменту конфігурацій програмного забезпечення.
7. Стандартизація внутрішніх процесів при виконанні програмних проектів.

Питання, що виносяться на самостійне вивчення студентом (3,5 год.):

1. Організація менеджменту конфігурацій при виконанні програмних проектів. [3, С. 183-198; 7, С. 1006]

9.1. Основні положення менеджменту конфігурації при реалізації програмних проектів

Сучасний процес розробки ПЗ являє собою складний творчій процес, який виконується у колективі. Менеджмент конфігурацій допомагає вирішити значну кількість невидимих, «закулісних» задач, які дозволяють розробникам ефективно функціонувати в команді.

Менеджмент конфігурації ПЗ – це організація компонентів програмної системи таким чином, щоб вони відповідали один одному у робочому процесі та функціонували синхронно.

Визначення за Роджером Прессманом: Менеджмент конфігурацій – сукупність операцій, призначених для контролю змін шляхом ідентифікації робочих продуктів, які, ймовірніше за все, були змінені, встановлення

взаємозв’язків між ними, визначення механізмів для управління різними версіями цих робочих продуктів, контролю призначених змін, а також перевірки і реєстрації змін».

Дане визначення важливе з того погляду, що найчастіше за все під менеджментом конфігурацій розуміють, насамперед, управління змінами.

Якісно організований менеджмент конфігурацій може вийти за рамки окремого програмного проекту і дозволити створити із програмних проектів актив, який може бути використаний багаторазово і із значною віддачею.

Менеджмент конфігурацій допомагає уникнути хаосу, подвійного технічного обслуговування, проблеми даних, що сумісно використовуються та модернізації продуктів.

Менеджмент конфігурацій можна представити у вигляді піраміди.

Рис. 9.1. Піраміда менеджменту конфігурацій

9.2.Контроль змін як одна із складових менеджменту конфігурацій

Як уже зазначалося, менеджмент конфігурацій дуже часто розглядається як, насамперед, контроль змін.

Важлива задача контролю змін полягає у тому, щоб врахувати зміни у елементи проекту одним учасником і довести їх до усіх інших учасників. При цьому слід врахувати можливість виникнення конфліктів змін, коли один і

той же елемент змінюється одночасно різними учасниками проекту, і невідомо, який саме варіант слід залишити, а який проігнорувати.

Контролю змін підлягають усі складові програмного проекту (які мають назву елементів конфігурації), зокрема: плани проекту, план управління ризиками, план забезпечення якості, безпосередньо сам план управління конфігураціями, специфікація системних вимог, системи показників, програмний код, результати тесту, інструменти, бібліотеки коду і сторонні компоненти та ін. Кількість взаємозв'язків між ними відображає складність задачі конфігурації.

9.3. Контроль версій як одна із складових менеджменту конфігурацій

Контроль версій – основна функція менеджменту конфігурацій при виконанні програмних проектів.

Контроль версій являє собою менеджмент версій програмного продукту як частина загального процесу менеджменту конфігурацій.

Основна задача контролю версій полягає у тому, щоб забезпечити одночасну роботу учасників проекту над спільним вихідним кодом.

Проблеми, які вирішує контроль версій:

- груповий синдром розробника – зусилля розробників можуть витрачатися даремно, якщо вони вносять модифікації у вихідний код, не погоджуючи їх один з одним;
- множинність версій – з багатьох причин часто буває потрібно одночасно підтримувати декілька версій програмного продукту, системи контролю версій дозволяють спростити цю задачу;
- сімейство програмних продуктів – забезпечення необхідності розробки продукту для різних платформ;
- можливість відслідковувати зміни вихідного коду і пов'язувати їх з вимогами і конкретними розробниками;
- недолік прозорості – у галузі ПЗ на відміну від матеріальної сфери, наприклад, будівництва дуже складно визначити рівень

завершеності роботи. Системи контролю версій дозволяють значно спростити вирішення цієї задачі;

- недолік контролю – процес розробки ПЗ досить складно контролювати, складно пов’язати обсяги виконаної роботи, роботи, що залишилася і встановлені бюджети та виділені ресурси. Системи контролю версій дозволяють встановити контроль над даними показниками;
- недолік трасування – відсутність зв’язків між окремими елементами і подіями проекту часто призводить до серйозних невдач, особливо із зростанням розміру проекту. Системи контролю версій дозволяють поєднувати елементи між собою і спростити такі процедури як, наприклад, внесення типових змін до різних версій проекту;
- недолік моніторингу – системи контролю версій є потужним інструментом різностороннього моніторингу процесу, наприклад, за допомогою систем контролю версій можна відслідкувати конкретний обсяг робіт, який був виконаний персоналом і визначити, наскільки виконана робота відповідає проектним вимогам;
- неконтрольовані зміни – неконтрольовані і непередбачені зміни є серйозним джерелом ризику для проекту, зміни можуть виникати за вимогою замовника чи з власної ініціативи розробника, однак їх неконтрольованість може бути причиною серйозних збоїв проекту і його повного провалу;
- атестація і верифікація – система контролю версій є вихідним інструментом, що дозволяє проводити атестацію і верифікацію програмного коду на відповідність вимогам.

9.4.Контроль випусків як одна із складових менеджменту конфігурацій

Контроль випусків – перетворення елементів конфігурації в завершений програмний продукт.

Сучасні системи управління конфігураціями дозволяють повністю автоматизувати процес «компіляція-збірка-тестування», в результаті якого можна отримати готову для використання версію програмного продукту.

Крім того, системи управління конфігураціями можуть бути інтегровані із системами підготовки дистрибутивів для того, щоб сформувати придатний для розповсюдження дистрибутив.

9.5.Поняття кортежу програми, кортежу проекту та паспорту проекту при виконанні програмних проектів

У процесі розробки програмного забезпечення управління конфігураціями тісно пов’язано з поняттям кортежу програми, можливий прийнятний варіант якого отримує назву «конфігурація».

Кортеж програми складається з наступних елементів:

- самої програми (у вихідних текстах або у машинних кодах);
- режиму експлуатації (включає вихідні дані та кваліфікацію користувача);
- середовища експлуатації (включає апаратну конфігурацію обладнання, системне програмне забезпечення, зокрема, операційну систему та драйвери обладнання, а також прикладне програмне забезпечення, яке може справляти вплив на роботу інших програм);
- документації, яка визначає допустиму множину вихідних даних, середовища, а також необхідну кваліфікацію користувачів програми.

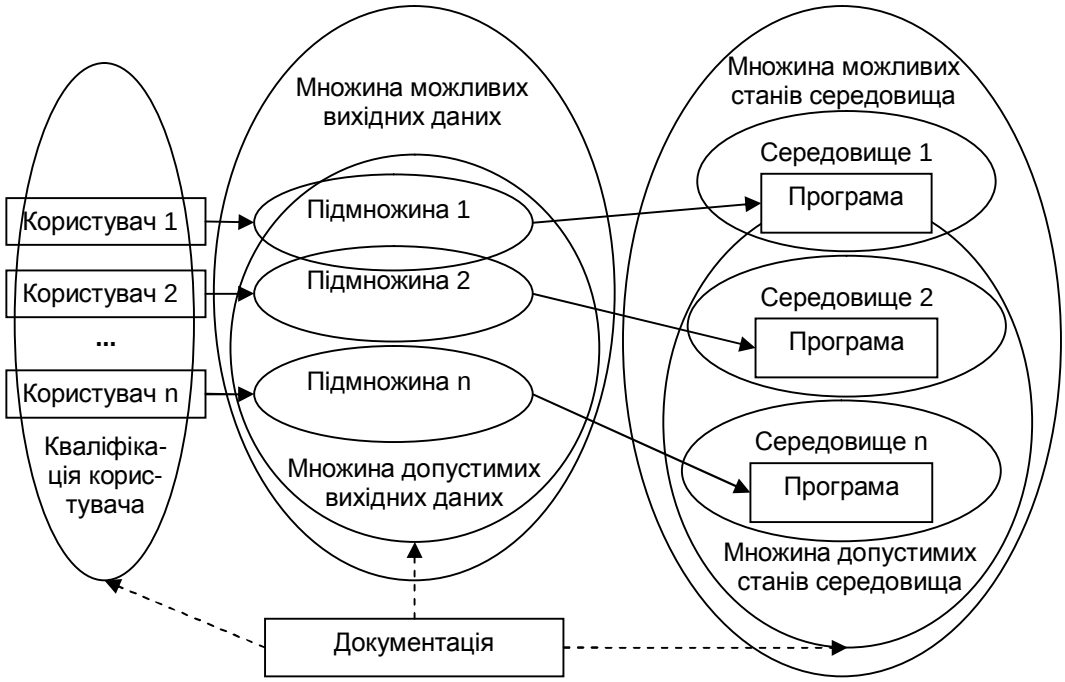

Рис. 9.2. Кортеж програми

Відповідно з рис. 9.2 інші складові кортежу, крім програмного коду та документації, такі як середовище виконання, вихідні дані та можлива кваліфікація користувачів можуть суттєво відрізнятися від одного випадку використання програмного забезпечення до іншого. Іноді ці складові мають непередбачувані та взагалі недопустимі значення, що, в кінцевому разі, негативно позначається на здатності програмного забезпечення функціонувати без збоїв, тобто відповідати критеріям якісної характеристики ПЗ, зокрема надійності.

Таким чином, надійність програмного забезпечення визначається не лише його програмним кодом, а усіма складовими кортежу програми, який являє собою конкретну реалізацію програмного забезпечення для конкретного режиму експлуатації та у конкретному середовищі. Документація обмежує допустиму множину вихідних даних, середовище експлуатації, а також кваліфікацію користувачів.

Аналізуючи кортеж програми, можна зробити висновок, що програмне забезпечення – це значно більше, ніж просто набір машинних кодів.

Наприклад, документація є невід'ємною складовою програмного забезпечення, адже саме вона визначає умови, для яких використання програмного забезпечення є допустимим, а для яких – ні. Некоректно написана документація рівноцінна помилці у програмному коді, адже вона не дозволяє використати систему для вирішення тієї чи іншої задачі. Крім того, за рахунок внесення змін до документації можна надзвичайно дешево (у порівнянні із іншими методами) позбутися помилок у програмному коді – наприклад, обмежуючи середовище чи режим експлуатації програмного забезпечення.

В результаті можна стверджувати, що надійність, як складова якості програмного забезпечення, забезпечується шляхом підтримки кортежу програми в такому стані, в якому функціонування програмного забезпечення відповідає критеріям надійності.

В питанні управління інноваціями зокрема, і в тому, що стосується управління змінами при розробці ПЗ взагалі слід використовувати термін «кортеж проекту» по аналогії з терміном «кортеж програми».

Використання кортежу проекту може бути корисним для вирішення проблеми поновлення версій використовуваного інструментарію та бібліотек коду.

Крім використання великої кількості сторонніх компонент, робота над сучасним проектом ведеться з використанням значної кількості інструментальних засобів, за допомогою яких розробляється проектна документація, створюється вихідний та машинний код, проводиться тестування і виправлення помилок, готується дистрибутив продукту та ін. Всі ці компоненти становлять безпосередній вплив на процес розробки програмного продукту і на відповідність його вимогам якості та, зокрема, надійності.

Особливого значення кортеж, який становлять ці компоненти, набуває в тому випадку, якщо над проектом працює значна кількість розробників. В даному разі невідповідність версій бібліотек стороннього коду, чи

інструментарію, що використовуються розробниками, може призвести до значних складнощів з узгодженням проекту. При збільшенні кількості учасників проекту, актуальність даної проблеми зростає.

Зафіксувати складові кортежу проекту, задокументувати його допустимі значення можна за допомогою відповідного документу, який має бути доступним для всіх учасників проекту і повинен мати силу наказу. У найпростішому випадку даний документ може мати вигляд таблиці, поділеної на розділи, кожен з яких відповідає елементам кортежу, для складних проектів даний документ може бути більш структурованим і розділеним на окремі, відносно незалежні частини.

На нашу думку доцільно назвати подібний документ «паспорт проекту», адже цей термін найбільше відповідає його призначенню – приводити у відповідність власне проект і середовище, в якому він проходить всі стадії розвитку.

На рис. 9.3 представлено графічне зображення поняття «кортеж проекту» та зазначено місце документу «паспорт проекту», що визначає його допустимі значення.

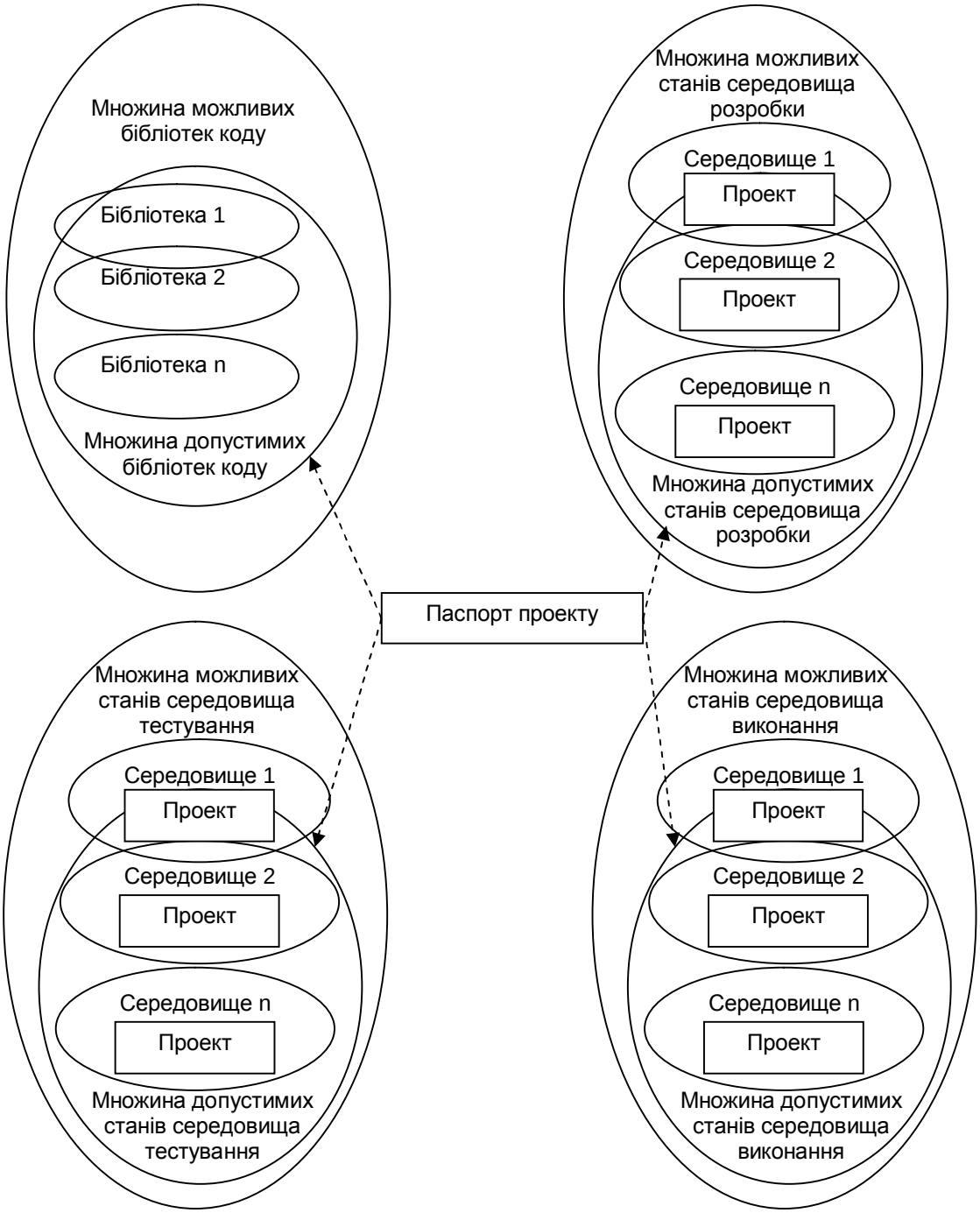

Рис. 9.3. Кортеж проекту

Розглянемо складові кортежу проекту, які визначаються паспортом проекту:

– допустимі бібліотеки коду – наводиться перелік бібліотек коду, які використовуються в проекті з обов’язковим зазначенням їх номерів версій та модифікацій, якщо такі є. Це можуть бути як сторонні бібліотеки коду, так і бібліотеки внутрішньої розробки;

– множина допустимих станів середовища розробки – дається перелік складових апаратно-програмного середовища, в якому ведеться робота над проектом, зокрема обов'язково вказуються номери версій інструментальних засобів, за допомогою яких здійснюється робота над окремими компонентами проекту, зазначається, чи використовуються обновлення та виправлення помилок;
– множина допустимих станів середовища тестування проекту – зазначається, в яких умовах та за допомогою якого інструментарію здійснюється перевірка функціональності проекту;
– множина допустимих станів середовища виконання проекту – зазначається, в яких умовах буде функціонувати кінцевий продукт. Ця складова кортежу проекту цілком відповідає складовій «середовище» кортежу програми.

Підтримувати паспорт проекту в актуальному стані та слідкувати за його дотриманням у команді, що працює над проектом, слід одному з керівників проекту, делегувати дані функції на більш низькі рівні управління, на нашу думку, недоцільно. Хоча делегування може бути здійснено, якщо робота виконується у великому колективі, в такому разі питанням підтримки паспорту проекту має займатися учасник проекту, за яким закріплено дані повноваження.

Паспорт проекту визначає такі ключові для проекту питання як:

– середовище виконання проекту – апаратні вимоги, вимоги до ОС, вимоги до мережного середовища, вимоги до СУБД й іншим зовнішнім по відношенню доданої системам, з якими повинні взаємодіяти отримані в результаті реалізації проекту прикладні програми;
– середовище розробки проекту – вказуються засоби автоматизованого проектування, використані мови програмування, компілятори й транслятори мов, інтегровані середовища програмування, засоби для проектування БД, у тому числі візуальні

конструктори запитів і дизайнери БД, використані компоненти, що не входять у стандартну поставку середовища програмування;

- засоби створення документації – використані інструменти для генерації документації, що входить у кінцеву поставку прикладного програмного забезпечення;
- засоби створення дистрибутивів – використане середовище для генерації дистрибутиву, що поставляється кінцевому користувачеві створеного у результаті реалізації проекту програмного продукту;
- середовище для тестування проекту – описуються середовище, а також інструментальні засоби, що застосовувалися для тестування проекту.

Під час реалізації великих проектів, у яких задіяне значне число розробників (у тому числі й залучених зі сторони), паспорт проекту дозволяє гарантувати, що всі вони працюють у єдиному середовищі, що виключає проблеми несумісності окремих інструментальних засобів, конфлікт версій, неможливість відтворення програмних помилок і т.д.

Паспорт проекту дозволяє відстежити проблеми, породжені інструментальними засобами й іншими засобами, створеними поза фірмою-розробником програмного забезпечення. Будь-які проблеми, що виникають із тим або іншим інструментом, або середовищем експлуатування готового програмного продукту документуються й приймаються заходи для їхнього усунення.

Окремі проблеми усуваються шляхом документації особливостей роботи тих або інших засобів, для інших – необхідно прийняти заходи для їхньої заміни.

Паспорт проекту є вихідною точкою для управління процесом впровадження інновацій у процесі роботи над проектом – будь-які зміни, що стосуються ключових питань розробки відбиваються в даному документі й повинні бути прийняті всіма учасниками проекту до уваги.

У випадку, якщо робота над проектом припиняється або відкладається на тривалий строк, використання документа паспорт проекту дозволяє при необхідності відтворити актуальне середовище розробки при поверненні до робіт над проектом.

З огляду на можливість роботи тих самих розробників над різними проектами, у процесі якої вони можуть використовувати різне середовище розробки, у тому числі й таке, що виключає одне одного, документ паспорт проекту дозволяє їм відтворити актуальне середовище розробки для роботи над кожним конкретним проектом. Також паспорт проекту може бути використаний розробником при початковій підготовці свого робочого місця до роботи над конкретним проектом.

Сам процес переходу на нові версії бібліотек коду та інструментарію має бути також впорядкованим і задокументованим. Його характер дуже сильно залежить від масштабу проекту та розміру компанії-розробника. Можливо, його слід проводити поетапно, чи з певною «тестовою» фазою, такою, як, наприклад, реалізація пілотного проекту.

Бажано, в процесі переходу на нову версію інструментарію, бібліотек коду чи будь-якої іншої складової кортежу проекту заборонити написання нового коду та розширення функціональності продукту за виключенням виправлення критично важливих помилок.

Також слід порекомендувати використовувати систему контролю версій компанією-розробником програмного забезпечення для збереження не лише вихідного коду проекту та пов’язаних з ним файлів, а й сторонніх бібліотек коду, що дозволить полегшити контроль за впровадженням нових версій використовуваних бібліотек коду розробниками, а також спростити підтримку «унаслідуваних» проектів, для яких перехід на нові версії бібліотек коду та інструментарію є недоцільним.

9.6. Інструментальні засоби менеджменту конфігурацій програмного забезпечення

Сучасні інструментальні засоби менеджменту конфігурацій стали логічним розвитком систем контролю версій. Починаючи з простих інструментів для контролю версій на рівні окремих файлів, що було необхідно, насамперед, для забезпечення взаємодії учасників проектів, системи контролю версій поступово отримали необхідні функціональні можливості систем менеджменту конфігурацій.

Популярні системи менеджменту конфігурацій, такі як Microsoft Team Server, Borland StarTeam реалізують повний спектр задач менеджменту конфігурацій і виступають у ролі центральної ланки, яка об'єднує навколо себе учасників програмних проектів.

Рекомендується ознайомитися із наступним матеріалом:

http://www.itc.ua/article.phtml?ID=21878

http://www.itc.ua/article.phtml?ID=22863

9.7. Стандартизація внутрішніх процесів при виконанні програмних проектів

Іншим важливим питанням при реалізації програмних проектів є питання стандартизації внутрішніх процесів компанії. Згідно проведеного Ф. Бруксом дослідження, якщо реальний графік виконання програмного проекту відстає від планових показників, то збільшення кількості людських ресурсів на пізніх етапах реалізації проекту лише погіршить ситуацію, і строки виконання проекту будуть відкладені на більш пізній термін.

Основна причина погіршення часових показників виконання проекту при збільшенні кількості доступних людських ресурсів для його виконання полягає у необхідності ознайомлення нових співробітників із особливостями реалізації проекту, що вимагає додаткових витрат часу як для нових учасників проекту, так і призводить до того, що відволікаються ресурси, які вже задіяні у реалізації проекту.

Важливим механізмом, який може сприяти значному скороченню неефективного використання робочого часу під час ознайомлення нових співробітників із особливостями реалізації проекту, а також зниженню неоднозначності у сприйнятті інформації як новими, так і існуючими учасниками проекту, має бути запровадження комплексу процедур і заходів, спрямованих на стандартизацію внутрішніх процесів.

Стандартизація підходів до створення програмного забезпечення в рамках організації, зокрема стандартизація написання програмного коду, дозволяє отримати наступні переваги:

- спрощується процес заміни персоналу, зокрема підготовки нового персоналу для участі у проекті, вивчення особливостей реалізації проекту;
- програмний код стає більш прозорим і зрозумілим, а також містить меншу кількість помилок за рахунок використання випробуваних на практиці і визнаних фахівцями підходів до його створення;
- зникає проблема «приватного володіння» програмним кодом, коли пояснювати деталі реалізації і вносити зміни до вихідного коду здатен лише розробник, який його створював;
- ефективність розробки ПЗ може суттєво зрости за рахунок зміни моделі володіння кодом – будь-який розробник може змінювати код будь-якого іншого учасника проекту, якщо визнає його недосконалим;
- забезпечується підтримка так званого «захищеного програмування», спрямованого на створення ПЗ таким чином, щоб на початку написання коду закласти механізми простого його розширення, пошуку різного виду помилок, налагодження і тестування, а також програмування з мінімальною кількістю помилок;
- суттєво спрощується процес управління розробкою ПЗ, в тому числі й інноваційною діяльністю за рахунок зростання прозорості процесу розробки.