У статті розглядаються питання проведення інженерного аналізу програмних засобів криптографічного захисту інформації, визначені фактори, що впливають на рівень безпеки їхнього функціонування, надані пропозиції щодо оцінки рівня безпеки програмних шифраторів.
Ключові слова: криптографія, рівень безпеки, програмні шифратори.

В статье рассмариваются вопросы проведения инженерного анализа программных средств киптографической защиты информации, определены факторы, влияющие на уровень безопасности их функционирования, наданы предложения по оценке программных шифраторов.
Ключевые слова: криптография, уровень безопасности, программные шифраторы.

The article is devoted to the questions of reverse engineering of a cryptographic data protection software. The factors that influence on their level of safety have been defined. The propositions to the evaluation of programmed cipherers have been made.
Key words: cryptography, level of safety, programmed cipherers.

Надійшла 27.01.2010

УДК 004.057.5
к.т.н., доц. Казакова Н.Ф. (МГУ, м.Одеса)

ОРГАНІЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИЩЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Постановка проблеми в загальному вигляді та її зв'язок з науковими і практичними завданнями. Програмне забезпечення – це та рушійна сила, яка сьогодні забезпечує функціонування торгівлі, промисловості, системи державного управління і зв'язує воєдино різні шари суспільства. Програмне забезпечення (в тому числі – для захищених інформаційних систем) допомагає створювати інформацію, надавати до неї доступ та візуалізувати її, причому все це великою множиною немислимих способів і в неймовірних формах. Саме приголомшливий прогрес в області програмного забезпечення допоміг справитися з розвитком світової економіки. На сьогоднішній день програмне забезпечення є невід'ємною частиною сучасного суспільства.

Звичайно ж, професійних розробників програмного забезпечення радує те, що світова економіка стає все більш залежною від програмного забезпечення. Завдяки розвитку техніки стало можливим створення програмних систем, які постійно ростуть, ускладнюються, розповсюджуються і стають все більш важливими. Крім того, росте ще й суспільний попит на ці системи.

Аналіз наукової і технічної літератури (наприклад [1...7]) показує, що збільшення складності, важливості і все більш широке розповсюдження програмних систем обмежують людські здібності до розуміння принципів розробки продуктів у сфері програмного забезпечення для захищених інформаційних систем. Спроба поліпшення існуючих систем в цілях їх адаптації до новітніх технологій приводить до виникнення ряду технічних та організаційних проблем. Ситуацію ускладнює ще й те, що комерційна галузь вимагає все більш ефективних та якісних продуктів, що розробляються і поширюються надзвичайно швидко. Крім того, на сьогодні розробники не в змозі повністю задовольнити існуючий попит.

Отже, на сьогоднішній день створення та технічна підтримка програмного забезпечення є надзвичайно важким та організаційно недосконалим процесом. Крім того, для створення якісного програмного забезпечення саме для захищених інформаційних систем важко синтезувати уніфікований процес, який можна повторити і результати дії якого можна

передбачити. У цій області слід зазначити дослідження G.Booch, K.Bittner, M.Ericsson, L.Probasco, S.Bylund, H.Dyrhage, J.Smith, J.Madhur і B.Katz (США) – співробітників міжнародної групи розробки програмного продукту Rational Process Development Group, – які в тому або іншому ступені враховують вище вказані проблеми. Крім того, вирішенням проблем у відміченій області займаються Ian.Gavin, I.Spence і M.Tudball (Великобританія), J.Hemphill (Франція) і P.Clarke (Швеція) та інші вчені. Це говорить про те, пошук нових ефективних методів організації процесу розробки програмного забезпечення для захищених інформаційних систем, придатних для практичного застосування, залишається поки *актуальним* і не вирішеним до кінця завданням.

Метою статті є акцентування уваги на суті та особливостях раціонального уніфікованого процесу (*Rational Unified Process*) організації розробки програмного забезпечення для захищених інформаційних систем.

Перейдемо до аналізу проблем процесу розробки програмного забезпечення для захищених інформаційних систем (далі – програмного забезпечення).

Різні проекти розробки програмного забезпечення часто терплять невдачу. У цьому сенсі можна виділити основні показники, які характеризують «провал» проектів:

- *Неточне розуміння потреб кінцевих користувачів;*
- *Нездатність роботи в умовах змінних вимог;*
- *Продукт складається з несумісних модулів;*
- *Програмне забезпечення важко підтримувати або розширювати;*
- *Пізнє виявлення істотних вад проекту;*
- *Погана якість програмного забезпечення;*
- *Неприйнятна продуктивність;*
- *Кожен співробітник займається чимось своїм, причому незрозуміло, хто, коли, де і навіщо що-небудь міняє;*
- *Ненадійний процес створення-випуску.*

На жаль, розпізнати симптом – ще не означає виявити хворобу. Наприклад, пізнє виявлення в проекті серйозної вади – це тільки показник великих проблем, які називають *суб'єктивною оцінкою стану проекту* і невиявленими суперечностями у вимогах, проектах і реалізаціях системи, яка розробляється. Хоча різні проекти «провалюються» з різних причин, схоже, що більшість невдач відбуваються унаслідок поєднання наступних основних причин:

- *Невміле управління спеціальними вимогами;*
- *Невизначений і неточний зв'язок;*
- *Ненадійна архітектура;*
- *Надмірна складність;*
- *Невиявлені суперечності у вимогах, проектах і реалізаціях;*
- *Недостатнє тестування;*
- *Суб'єктивна оцінка стану проекту;*
- *Нездатність справитися з ризиком, який може проявитися з часом;*
- *Некероване розповсюдження змін;*
- *Недостатня автоматизація.*

Розглянемо методи оптимізації при організації процесу розробки програмного забезпечення.

Своєчасне розпізнавання причини дозволяє не тільки усунути її симптоми, але й зайняти більш виграшну позицію з погляду розробки та підтримки якісного програмного забезпечення за допомогою процесу, який можна повторити, а результати дії якого можна

передбачити. Методи оптимізації при організації процесу розробки програмного забезпечення, що потрапили в огляд, – це комерційно доведені підходи до розробки програмного забезпечення, які, будучи використані разом, усувають основні причини проблем, які виникають при розробці програмного забезпечення. Вони є «кращими» не тому, що можна точно визначити їх цінність, а швидше тому, що їх широко використовують процвітаючі організації. Отже, пропоновані методи по організації виробничих процесів при розробці програмного забезпечення, можна сформулювати таким чином:

- *Ітеративна розробка;*
- *Управління вимогами;*
- *Використання модульної архітектури;*
- *Використання візуального моделювання;*
- *Систематична перевірка якості;*
- *Спостереження за змінами;*

Класичний підхід до розробки програмного забезпечення включає життєвий цикл у вигляді водоспаду (рис. 1). При такому підході розробка послідовно проходить фази аналізу вимог, проектування, програмування і тестування елементів, а також тестування підсистеми і всієї системи в цілому.

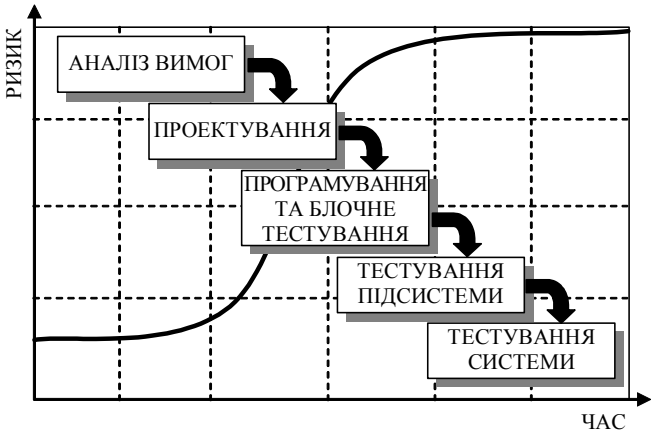

Рис.1. Водоспадний життєвий цикл

Основною проблемою описаного підходу є зростання ризику з часом. Це припускає виникнення проблеми усунення помилок попередніх фаз: процес стає дуже тривалим. Первинний проект, напевно, матиме вади в ключових вимогах, і, більш того, пізнє виявлення проектних недоробок може привести до перевитрати засобів або відмови від проекту. Підхід у вигляді «водоспаду» має тенденцію до «маскування» дійсних ризиків до тих пір, поки надто пізно на них не реагуватиме користувач. Т.ч., приходимо до висновку про те, що альтернативою підходу типу «водоспад» є поелементний ітеративний процес (рис. 2).

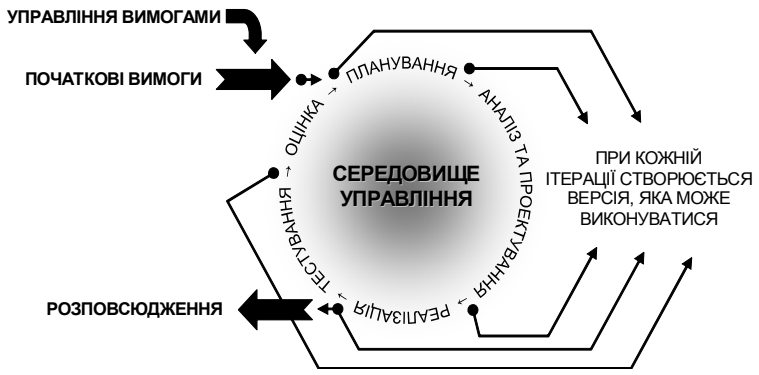

Рис. 2. Поелементний ітеративний процес

При такому підході, заснованому на спіральній моделі Баррі Боєма (Barry Boehm), встановлення рівня ризику, загрозливого проекту, здійснюється на раніших етапах життєвого циклу, коли ще можна ефективно і своєчасно реагувати на них. За допомогою даного підходу здійснюється безперервне виявлення, дослідження і реалізація, причому при кожній ітерації група розробників управляє артефактами проекту в цілях їх поступового зближення.

Ітеративний підхід має безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення:

- *Істотні непорозуміння стають очевидними на ранніх етапах життєвого циклу, коли ще можна прийняти заходи по їх усуненню;*
- *Описаний підхід сприяє встановленню зворотного зв'язку з користувачем в цілях з'ясування дійсних вимог до системи;*
- *Група розробників концентрує свою увагу на особливо важливих питаннях проекту, а не на тих, які тільки відволікають увагу команди від дійсно небезпечних моментів;*
- *Безперервне ітеративне тестування дозволяє об'єктивно оцінити стан проекту;*
- *Суперечності у вимогах, проектах і реалізаціях виявляються раніше;*
- *Навантаження команди (особливо команди тестування) зростає у міру розвитку проекту;*
- *Команда може навчатися і внаслідок цього безперервно покращувати процес;*
- *Впродовж всього життєвого циклу проекту його організатори можуть отримувати реальне уявлення про поточний стан проекту.*

Наступною проблемою при організації процесу розробки програмного забезпечення, є питання управління вимогами. Складність управління вимогами програмних систем полягає в динамічності цих вимог: варто очікувати, що в процесі розвитку проекту вони зміняться. Більш того, визначення дійсних вимог системи, необхідних для досягнення економічних і технічних цілей системи, є безперервним процесом. Цілком визначити системні вимоги до початку процесу розробки можна тільки для тривіальних систем. Дійсно, при видозмінах нової або модернізованої системи міняється і розуміння користувачем вимог до неї. Взагалі, *вимога* – це умова або характеристика, якій повинна відповідати система. Активне управління вимогами включає три види діяльності:

1. *Встановлення, впорядкування і документування функціональних можливостей і обмежень системи;*
2. *Оцінка змін вимог і визначення їх впливу на систему;*
3. *Відстежування і документування рішень і компромісів.*

Управління вимогами проекту пропонує безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення:

- *Управління вимогами дозволяє упорядкувати процес розробки програмного забезпечення;*
- *Зв'язок ґрунтується на певних вимогах;*
- *Вимоги можуть розташовуватися відповідно до пріоритетів; крім того, їх можна відфільтрувати і відстежити;*
- *Можлива об'єктивна оцінка функціональних можливостей і продуктивності;*
- *Суперечності виявляються на ранніх етапах проекту;*
- *При належній інструментальній підтримці можна створити корпоративну базу даних системних вимог, параметрів і відстежування, яка автоматично пов'язується із зовнішніми документами.*

Візуалізація, специфікація, створення та документування програмної системи – все це залежить від виду системи з різних точок зору. Кожна з зацікавлених сторін – кінцевих користувачів, аналітиків, розробників, системних інтеграторів, випробувачів, технічних редакторів і керівників проекту – привносить своє бачення проекту, причому кожен з них бачить систему з якоїсь своєї точки зору, яка, до того ж, може мінятися в процесі розвитку проекту. Тому системна архітектура – це, ймовірно, найважливіший компонент, який може

використовуватися для управління цими різними точками зору і, у зв'язку з цим, управляти нарощуваним ітеративним розвитком системи під час її життєвого циклу.

Системна архітектура включає рішення відносно:

- *Організації програмної системи;*
- *Вибору структурних елементів та їх інтерфейсів при формуванні системи;*
- *Поведінки цих структурних елементів, яка обумовлена їх спільною роботою;*
- *Формування із структурних елементів і ліній їх поведінки підсистем, що поступово укрупняються;*
- *Архітектурного стилю, що координує структуру, яка складається з елементів і їх інтерфейсів, а також їх спільної роботи і об'єднання.*

Архітектура програмного забезпечення пов'язана не тільки з питаннями структури і поведінки, але й з питаннями використання, функціональних можливостей, продуктивності, еластичності, повторного використання, легкої зрозумілості, економічних і технологічних обмежень, компромісів, а також питаннями естетики.

Еластичність архітектури характеризується економічною можливістю повторного використання системи, що, у свою чергу, дозволяє очевидним чином розділити роботу між командами розробників, виділити залежності від апаратного і програмного забезпечення, які можуть піддаватися змінам, і зробити експлуатацію зручнішою.

Важливим підходом до архітектури програмного забезпечення для захищених інформаційних систем є модульна розробка, що отримала назву CBD (*Component Based Development*), дозволяє повторно використовувати або настроювати компоненти з множини комерційно доступних джерел. Модель компонентних об'єктів COM (*Component Object Model*) корпорації Microsoft, архітектура CORBA (*Common Object Request Broker Architecture*) групи Object Management Group і Enterprise Javabeans (EJB) від компанії Sun Microsystems – приклади поширених і широко підтримуваних платформ на яких може використовуватися модульна архітектура. Як показано на рис. 3, який достатньо відомий з літературних джерел, наприклад [8], компоненти роблять можливим повторне використання в значних масштабах, дозволяють системам складатися з існуючих частин, готових частин від сторонніх виробників і нових, які відносяться до певних областей і об'єднують решту частин.

Ітеративна розробка програмного забезпечення з використанням модульної архітектури пов'язана з безперервним розвитком архітектури системи. При кожній ітерації створюється архітектура, що реалізується, яку можна протестувати і оцінити по відношенню до системних вимог. Крім того, цей підхід дозволяє команді розробників безперервно боротися з найважливішими ризиками проекту. Використання модульної архітектури пропонує безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення:

- *Модулі сприяють еластичності архітектури;*
- *При внесенні змін модульна структура дозволяє явно розділити обов'язки між елементами системи;*
- *Завдяки стандартизованим конструкціям (таким, як Com+, CORBA і EJB) і комерційно доступним компонентам можливе повторне використання системи;*
- *На основі модулів природним чином організовується управління конфігурацією;*
- *Засоби візуального моделювання автоматизують модульну розробку.*

Моделювання – це важливо, оскільки воно допомагає команді розробників візуалізувати, визначати, створювати і документувати структуру та поведінку архітектури системи. Використання стандартної мови моделювання, наприклад, UML (*Unified Model Language* – уніфікована мова моделювання), дозволяє членам команди розробників однозначно доносити свої рішення один одному.

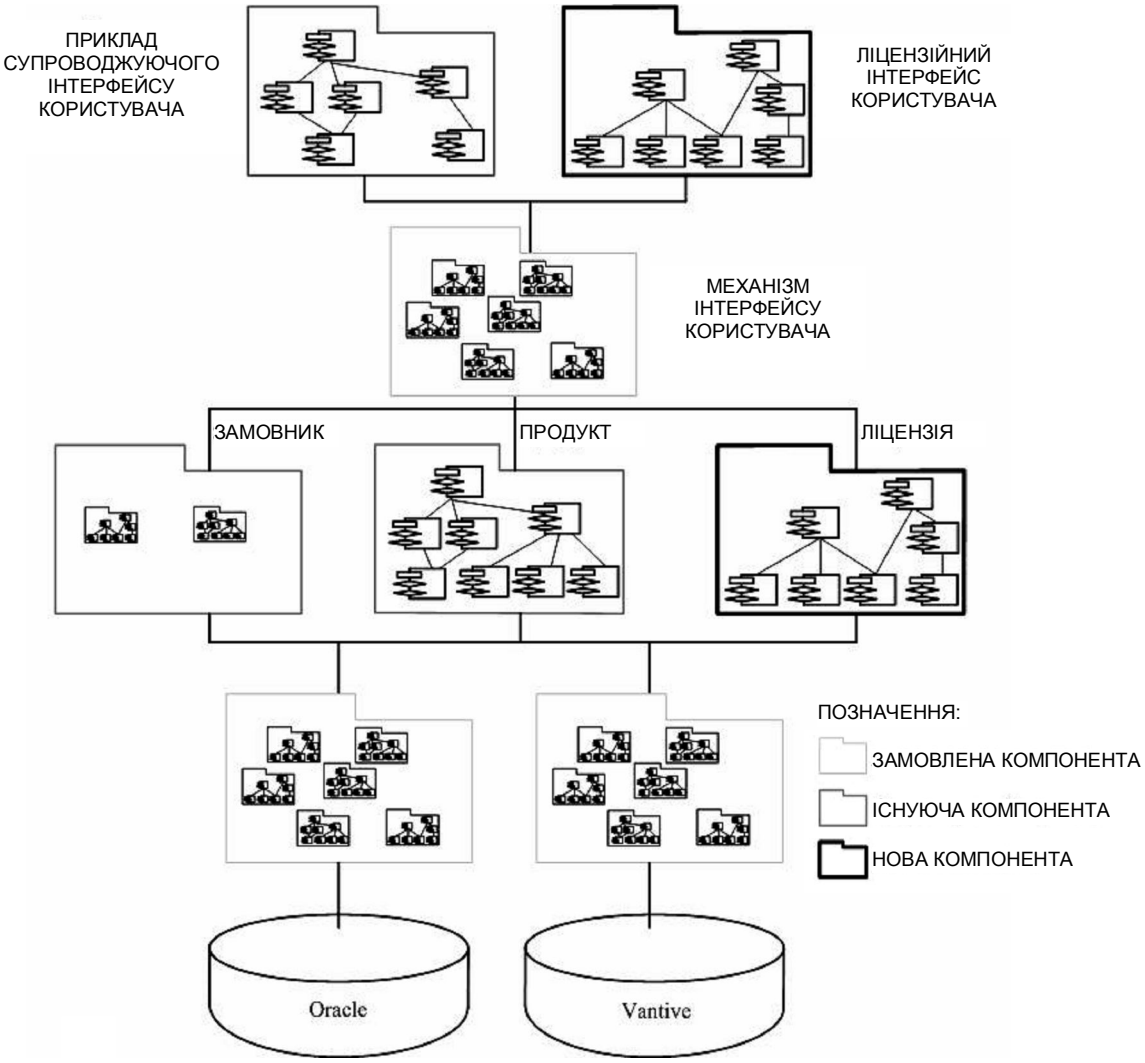

Рис. 3. Модульна архітектура програмного забезпечення [8]

Засоби візуального моделювання полегшують управління моделями, дозволяючи показувати або приховувати подробиці у міру потреби. Візуальне моделювання також сприяє підтримці несуперечності артефактів системи: вимог, проектів і реалізацій. Іншими словами, візуальне моделювання дає команді можливість боротися з складністю програмного забезпечення.

Застосування ітеративної розробки з використанням візуального моделювання дозволяє виділяти і оцінювати зміни в архітектурі, а також повідомляти про них всій команді розробників, та згодом, під час кожної ітерації можна, за наявності належних засобів, синхронізувати моделі з початковими кодами.

Модель – це спрощення дійсності, яке повністю описує систему з певної точки зору, як показано на рис. 4. Модель допомагає краще зрозуміти модульовану систему. Крім того, в складних системах, які неможливо представити повністю, без моделі обійтися взагалі неможливо.

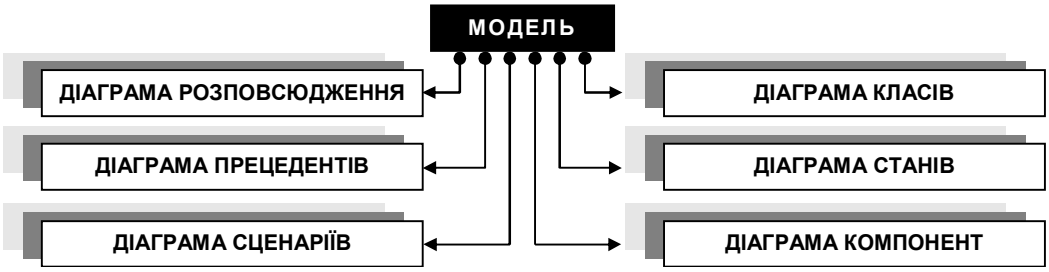

Рис. 4. Упрощение действительности, которое описывает систему с определенной точки зрения

Візуальне моделювання програмного забезпечення пропонує безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення для захищених інформаційних систем:

- *Прецеденти і сценарії однозначно визначають лінії поведінки;*
- *Моделі однозначно фіксують структуру програмного забезпечення;*
- *Виявляється немодульна і нееластична архітектура;*
- *За необхідністю можна приховати подробиці;*
- *У однозначних проектах явніше видно суперечності;*
- *Якість додатку починається з хорошого проекту;*
- *Засоби візуального моделювання підтримують моделювання на мові UML.*

Як показано на рис. 5, пошук і виправлення помилок після розповсюдження програмного забезпечення обходиться в 100...1000 разів дорожче, ніж до розповсюдження. З цієї причини важливо постійно визначати *якість системи*, використовуючи як критерії її функціональні можливості, надійність та ефективність.

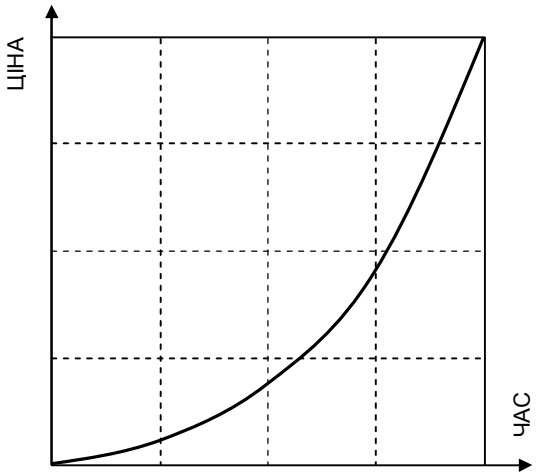

Рис. 5. Ціна виправлення помилки

Перевірка функціональних можливостей системи – основна частина роботи по тестуванню – включає створення тестів для всіх ключових сценаріїв, кожен з яких представляє деякий аспект очікуваної поведінки системи. Визначити функціональні можливості системи можна, з'ясувавши, які сценарії «зірвалися», де це відбулося і які сценарії та відповідні коди ще не були виконані. Якщо програмне забезпечення розробляється ітеративно, то й тестування слід проводити при кожній ітерації, тобто необхідно проводити безперервну якісну оцінку.

Перевірка якості програмного забезпечення пропонує безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення:

- *Стан проекту оцінюється об'єктивно, а не суб'єктивно, оскільки оцінюються результати тестів, а не те, що написане на папері;*
- *Ця об'єктивна оцінка показує суперечності у вимогах, проектах і реалізаціях;*
- *Тестування і контроль зосереджені в областях найбільшого ризику і, отже, в цих областях підвищується якість і ефективність системи;*

- *Дефекти виявляються на ранніх етапах, що значно знижує вартість їх виправлення;*
- *Автоматизовані засоби тестування дозволяють тестувати функціональні можливості, надійність і продуктивність системи.*

Основною проблемою при розробці програмних систем, які відносяться до сфери захисту інформаційних ресурсів, є необхідність роботи з деякою множиною команд розробників, що знаходяться, можливо, в різних місцях та мають відповідні рівні допусків до роботи з закритою інформацією. Всі вони спільно працюють з множинними ітераціями, версіями, продуктами і платформами. За відсутності строгого контролю та координації їх роботи процес розробки, у зв'язку з зазначеним обмеженням, швидко стане хаотичним.

Координація дій і програмного забезпечення, що створюються окремими розробниками і цілими командами, включає визначення технологічних процесів, направлених на управління змінами, що вносяться до програмного забезпечення і інших виробничих чинників. Це дозволяє краще розподілити ресурси, ґрунтуючись на пріоритетах і ризиках проекту. Крім того, за допомогою координації можна активно управляти роботою над змінами в різних ітераціях. Такий підхід, разом з ітеративною розробкою програмного забезпечення, дозволяє безперервно спостерігати за змінами, що сприяє своєчасному виявленню проблеми і швидкому реагуванню на неї.

Координація ітерацій та версій включає встановлення і випуск перевіреної базової лінії в кінці кожної ітерації (рис. 2). Крім того, для оцінки впливу змін і активного управління ними необхідне існування можливості оперативного контролю над елементами кожної версії і однаковими елементами множинних паралельних версій.

Контроль над змінами в програмному забезпеченні пропонує безліч рішень, що знімають основні причини проблем в процесі розробки програмного забезпечення:

- *Результати кожної дії технологічного процесу управління вимогами можна передбачити, а сам процес повторити;*
- *Запити на внесення змін сприяють встановленню надійнішого зв'язку між групами розробників і зацікавленими сторонами;*
- *Робота у відособлених закритих групах усуває взаємні перешкоди, що виникають між співробітниками, які паралельно виконують яке-небудь завдання;*
- *Збір статистичних даних дозволяє об'єктивно оцінити стан проекту;*
- *Робочі середовища містять всі параметри, що погоджують процес;*
- *Розповсюдження змін піддається оцінці та контролю;*
- *Зміни можна підтримувати в стійких системах, що настроюються.*

На закінчення приведемо анотацію раціонального уніфікованого процесу розробки програмного забезпечення (*Rational Unified Process*) [8], який знайшов широке розповсюдження серед зарубіжних розробників. Суть його заключається в наступному:

Існує чотири функції процесу розробки програмного забезпечення:

1. *Забезпечення керівництва послідовністю дій команди;*
2. *Визначення, які параметри повинні створюватися і коли;*
3. *Конкретні відомості стосовно того, чим повинні займатися окремі розробники і цілі команди;*
4. *Пропонування критеріїв спостереження та вимірювання продуктів і видів діяльності проекту.*

Без системно організованого процесу розробки програмного забезпечення для захищених інформаційних систем команда розробників, спілкування яких обмежене режимними заходами, працюватиме епізодично, а успіх залежатиме від героїчних зусиль окремих співробітників. Такий стан ніяк не можна назвати задовільним. І навпаки, системна координація їх роботи надає можливості організаціям-розробникам розробляти складні системи за допомогою процесу, результати кожної дії якого можна передбачити, а сам процес повторити кожним з виконавців. Причому сказане справедливо не тільки для сталих операцій, але і для тих, які можуть поліпшуватися з кожним новим проектом, підвищуючи ефективність і продуктивність структури програмного забезпечення в цілому.

Подібні однозначні процеси сприяють використанню всіх порад, запропонованих вище. Коли всі ці поради будуть враховані в процесі, команда розробників зможе застосувати колективний досвід використання тисяч успішних проектів.

Висновки

Достатньо складно створити якісне програмне забезпечення за допомогою процесу, результати кожної дії якого можна передбачити, а сам процес повторити. Існує декілька показників поширених проблем розробки програмного забезпечення, що виникають унаслідок *певних основних причин*. Усунути основні причини проблем, що виникають при розробці програмного забезпечення для захищених інформаційних систем, можна, слідуючи запропонованим шести методам: ітеративна робота розробників; управління вимогами; використання модульної архітектури програмного забезпечення; використання візуального моделювання; контроль якості програмного продукту; контроль та координація змін.

Список літератури

1. Booch G., Kansas L. IEEE Software 15(1), January-February 1998, pp. 32-35 / [Електронний ресурс]: http://linkinghub.elsevier.com/retrieve/pii/S0377221702005337.

2. Jones C. Patterns of Software Systems Failure and Success. London: International Thompson Computer Press, 1996 / [Електронний ресурс]: http://www.amazon.com/Patterns-Software-System-Failure-Success/dp/1850328048.

3. Yourdon E., March D. Managing «Mission Impossible» Projects. Upper Saddle River, NJ: Prentice-Hall, 1997 / [Електронний ресурс]: http://www.student.cs.uwaterloo.ca/~cs446/F99/slides/9-deathmarch.ppt.

4. Сети Software Program Manager's Network / [Електронний ресурс]: http://www.spmn.com.

5. Gilb T. Principles of Software Engineering Management. Harlaw, UK: Addison-Wesley, 1988, p.73 / [Електронний ресурс]: http://pioneer.chula.ac.th/~sperapho/files/class/660/660_doc.pdf.

6. Barry W. Boehm. A Spiral Model of Software Development and Enhancement. IEEE Computer, May 1988, pp. 61-72 / [Електронний ресурс]: http://portal.acm.org/citation.cfm?id=12948.

7. Booch G. Object Solutions: Managing the Object-Oriented Project. Reading, MA: Addison-Wesley, 1995 / [Електронний ресурс]: http://www.mypearsonstore.com/bookstore/product.asp?isbn=0805305947&xid=PSED.

8. Кратчен Ф. Введение в Rational Unified Process. – 2-е изд. – 2002 / [Електронний ресурс]: http://bsu.iba.by.

Проведений аналіз і розкрита суть раціонального уніфікованого процесу організації розробки програмного забезпечення для захищених інформаційних систем.

Ключові слова: захищені інформаційні системи, розробка програмного забезпечення, модульна архітектура програмного забезпечення.

Проведен анализ и раскрыта суть рационального унифицированного процесса организации разработки программного обеспечения для защищенных информационных систем.

Ключеве слова: защищенные информационные системы, разработка программного обеспечения, модульная архитектура программного обеспечения.

The analysis and revealed the essence of Rational Unified Process organization of software development for secure information systems.

Key words: protected information systems, software engineering, software modular architecture.

Надійшла 24.02.2010