**Лекція 2 Моделі та стандарти конструювання програмного забезпечення**

**Моделі конструювання програмного забезпечення**.

Моделі конструювання визначають комплекс операцій, які включають послідовність, результати (наприклад, вихідний код та відповідні unit-тести) та інші аспекти, пов’язані з загальним життєвим циклом розробки програмного забезпечення. У більшості випадків моделі конструювання визначаються стандартом життєвого циклу, який використовується, методологіями та практиками, що застосовуються. Деякі стандарти життєвого циклу, по своїй природі, орієнтовані на конструювання – наприклад, екстремальне програмування. Деякі розглядають конструювання у нерозривному зв’язку з проектуванням (у частині моделювання), наприклад Rational Unified Process (RUP).

Створено багато моделей розробки програмного забезпечення. Ряд із них у більшій мірі сфокусований на конструюванні програмного забезпечення, як такому. Деякі моделі являються більш лінійними з точки зору конструювання ПЗ. До них відносяться, наприклад, водоспадна (waterfall) та поетапна (stage-delivery) моделі життєвого циклу. Ці моделі розглядають конструювання як діяльність, яка починає проводитись лише після завершення певних обов’язкових до виконання (prerequisite) робіт, що включають детальний опис вимог, докладний дизайн та детальне планування. Більш лінійні підходи намагаються підкреслити дії, яким передує конструювання і створити чіткіший розподіл між такими різними видами діяльності. У таких моделях основним змістом конструювання може бути кодування.

Інші моделі більш ітеративні, до них відносяться еволюційне прототипування, екстремальне програмування і Scrum. Ці підходи сходяться до розгляду конструювання як діяльності, що ведеться одночасно з іншими видами робіт по створенню ПЗ та пересікається з ними, включаючи визначення вимог, проектування і планування. Ці підходи змішують проектування, кодування і тестування, часто розглядаючи конструювання як їх комбінацію.

S**crum** (чит.як *скрам*) — підхід управління проектами для [гнучкої розробки програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%93%D0%BD%D1%83%D1%87%D0%BA%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). Scrum чітко робить акцент на якісному контролі процесу розробки.

Відповідно те, що саме розуміється під «конструюванням» залежить певною мірою від моделі життєвого циклу, що використовується.

**Стратегії конструювання ПЗ**

Існують 3 стратегії конструювання ПЗ:

- *однократний прохід* (водоспадна стратегія) — лінійна послідовність етапів конструювання;

- *інкрементна стратегія.* На початку процесу визначаються всі користувацькі та системні вимоги, решта конструювання виконується у вигляді послідовності версій. Перша версія реалізує частину запланованих можливостей, наступна версія реалізує додаткові можливості й т. д., поки не буде отримано повну систему;

- *еволюційна стратегія.* Система також будується у вигляді послідовності версій, але на початку процесу визначені не всі вимоги. Вони уточнюються в результаті розробки версій.

Характеристики стратегій конструювання ПЗ відповідно до вимог стандарту IEEE/EIA 12207.2 наведені у таблиці 2.1.

*Таблиця 2.1.*

|  |  |  |  |
| --- | --- | --- | --- |
| Стратегія конструювання | Чи всі вимоги визначені на початку процесу? | Декілька циклів конструювання? | Проміжне ПЗ розповсюджується? |
| Однократний прохід | Так | Ні | Ні |
| Інкрементна (заплановано покращення продукту) | Так | Так | Може бути |
| Еволюційна | Ні | Так | Так |

**Моделі систем. Прототипування програмних систем**

**Модель ПЗ** – **це формалізований опис системи ПЗ на певному рівні абстракції.** Кожна модель визначає конкретний аспект системи, використовує набір діаграм та документів заданого формату, а також відбиває точку зору та є об’єктом діяльності різних людей з конкретними інтересами, ролями або задачами. **Графічні (візуальні) моделі –** це засоби для візуалізації, опису, проектування та документування архітектури системи. **Моделі є** основою взаємодії учасників проекту і гарантують коректність архітектури. Оскільки складність систем зростає, важливо мати гарні методи моделювання. Хоча й існує безліч факторів, від яких залежить успіх проекту, але наявність суворого стандарту мови моделювання є вельми істотним.

**Склад моделей,** використовуваних в кожному конкретному проекті, та ступінь їх деталізації в загальному випадку залежать **від наступних факторів:**

- складності проектованої системи;

- необхідності повноти її опису;

- знань та навичок учасників проекту;

- часу, відведеного на проектування.

**Візуальне моделювання** сильно вплинуло на розвиток технологій проектування ПЗ взагалі та CASE-засобів зокрема.

**CASE-технологія** – це сукупність методів проектування ПЗ, а також набір інструментальних засобів, які дозволяють наочно моделювати предметну галузь, аналізувати цю модель на всіх стадіях розроблення і супроводу ПЗ, а також розробляти додатки відповідно до інформаційних потреб користувача на основі аналізу специфікацій.

**При структурному аналізі та проектуванні використовуються різні моделі, які описують:**

- функціональну структуру системи;

- послідовність виконуваних дій;

- передачу інформації між функціональними процесами;

- відношення між даними.

**Найбільш поширеними моделями перших трьох груп є:**

- функціональна модель SADT (Structured Analysis and Design Technique);

- модель IDEF3;

- діаграми потоків даних DFD (Data Flow Diagrams).

**Метод SADT** – це сукупність правил та процедур, призначених для побудови функціональної моделі об’єкту будь-якої предметної галузі. Функціональна модель SADT відображає функцій ну структуру об’єкту, тобто дії, які він виконує, та зв’язки між цими діями. Моделі SADT (IDEF0) традиційно використовуються для моделювання організаційних систем (бізнес-процесів). Слід зазначити, що метод SADT успішно працює лише при описі добре специфікованих та стандартизованих бізнес-процесів у корпораціях, тому прийнятий у США в якості типового.

**Перевагами застосування моделей SADT є**: повнота опису бізнес-процесу, жорсткі вимоги методу (як результат – моделі стандартного вигляду), відповідність підходу до опису процесів стандартам ISO 9000. Оскільки на Україні бізнес-процеси почали формуватись і розвиватись порівняно недавно, вони слабко типізовані, тому розумніше орієнтуватись на менш жорсткі моделі.

**Метод моделювання IDEF3** призначений для таких моделей процесів, в яких важливо зрозуміти послідовність виконання дій та взаємозалежності між ними. Основою моделі IDEF3 є так званий сценарій процесу, який виділяє послідовність дій та підпроцесів аналізованої системи.

**Діаграми потоків даних (DFD)** – це ієрархія функціональних процесів, зв’язаних потоками даних. Мета такого представлення – продемонструвати, як кожний процес перетворює свої вхідні дані у вихідні, а також виявити залежності між цими процесами. Модель системи визначається як ієрархія діаграм потоків даних, які описують асинхронний процес перетворення інформації від її ведення у систему до видачі результатів споживачу. **Діаграми потоків даних** спочатку створювались як засіб проектування інформаційних систем (SADT – засіб моделювання систем взагалі) і мають більший набір елементів, які адекватно відображають специфіку таких систем.

Розглянуті моделі приблизно однакові з точки зору можливостей зображувальних засобів моделювання. Основним критерієм вибору того чи іншого методу є ступінь володіння ним з боку консультанта або аналітика, грамотність вираження своїх думок мовою моделювання.

Найбільш поширеним засобом моделювання даних (предметної галузі) є модель «сутність-зв’язок» (Entity-Relationship Model - ERМ). Ця модель представляє собою підмножину об’єктної моделі предметної галузі.

Концептуальною основою об’єктно-орієнтованого аналізу та проектування (ООАП) ПЗ є об’єктна модель. Її основні принципи – абстрагування, інкапсуляція, модульність та ієрархія; основні поняття – об’єкт, клас, атрибут, операція, інтерфейс.

Більшість сучасних методів ООАП засновані на використанні уніфікованої **мови моделювання UML (Unified Modeling Language).**

**Мову UML використовують всі крупні компанії-виробники ПЗ (Microsoft, Oracle, IBM, Hewlett-Packard, Sybase).** Крім того, практично всі світові виробники CASE-засобів підтримують UML у своїх продуктах. Повний опис UML можна знайти на сайтах http://www.omg.org та http://www.rational.com.

**Стандарт UML містить наступний набір діаграм:**

**1) структурні (structural) моделі:**

- діаграми класів (class diagrams);

- діаграми компонентів (component diagrams);

- діаграми розташування (deployment diagrams);

**2) поведінкові (behavioral) моделі:**

- діаграми варіантів використання (use case diagrams);

- діаграми взаємодії (interaction diagrams): діаграми послідовності (sequence diagrams) та кооперативні діаграми (collaboration diagrams);

- діаграми станів (statechart diagrams);

- діаграми діяльності (activity diagrams).

UML має механізми розширення, призначені для можливості адаптації мови моделювання розробниками до своїх конкретних потреб, не змінюючи при цьому його метамодель. **Наявність механізмів розширення принципово відрізняє UML від таких засобів моделювання, як IDEF0, IDEF1X, IDEF3, DFD та ERM.** Перераховані мови є сильно типізованими, оскільки вони не допускають довільної інтерпретації семантики елементів моделей. UML, допускаючи таку інтерпретацію, є слабко типізованою мовою.

**До його механізмів розширення належать:**

- **стереотипи** – новий тип елементу моделі, який визначається на основі вже існуючого елементу; це механізм, який дозволяє розділяти класи на категорії;

- **теговані** (іменовані) значення – це пара рядків «тег=значення» або «ім’я=вміст», в яких зберігається додаткова інформація про будь-який елемент системи;

- **обмеження** – це семантичне обмеження, яке має вигляд текстового виразу природньою або формальною мовою, яке неможливо виразити мовою UML.

Моделювання бізнес-процесів є важливою складовою частиною проектів по створенню великомасштабних систем ПЗ. Відсутність таких моделей є однією з головних причин невдач багатьох проектів. Моделі бізнес-процесів є самостійним результатом з великим практичним значенням. Наразі у моделюванні бізнес-процесів переважає процес ний підхід. Його основний принцип полягає у структуруванні діяльності організації відповідно до її бізнес-процесів, а не організаційно-штатної структури. Процесний підхід може використовувати будь-які з вище перерахованих засобів моделювання. Однак наразі спостерігається тенденція інтеграції різноманітних методів моделювання та аналізу систем, яка проявляється у формі створення інтегрованих засобів моделювання.

**Одним з таких засобів є продукт під назвою ARIS -** Architecture of Integrated Information System – це комплекс засобів аналізу та моделювання діяльності підприємства на основі сукупності різних методів моделювання.

**ARIS підтримує чотири типи моделей:**

організаційні моделі (структура системи);

функціональні моделі (ієрархія цілей системи);

інформаційні моделі (структура інформації системи);

моделі керування (комплексний погляд на реалізацію бізнес-процесів в межах системи).

Під час моделювання кожний аспект діяльності підприємства спочатку розглядається окремо, а після детального припрацювання всіх аспектів будується інтегрована модель, яка відбиває всі зв’язки між різними аспектами.

Моделювання бізнес-процесів передбачає побудову двох моделей: **модель бізнес-процесів (Business Use Case Model) та модель бізнес-аналізу (Business Analysis Model).** Для моделювання бізнес-процесів можуть бути використані діаграми мови UML, але при моделюванні діяльності великої компанії, яка не лише виробляє продукцію, але й надає послуги, необхідно застосовувати різні методики моделювання.

**Прототипування ПЗ –** це етап розроблення ПЗ, процес створення прототипу програми – макету (чорнової, пробної версії) програми, як правило, з метою перевірки придатності пропонованих для застосування концепцій, архітектурних та/або інших технологічних рішень, а також для представлення програми замовнику на ранніх етапах процесу розроблення.

**Прототип дозволяє також одержати зворотній зв’язок від майбутніх користувачів, причому, саме тоді, коли це найбільш необхідно:** на початку проекту, коли ще є можливість виправити помилки проектування практично без втрат. Прототипування спрямовано на перевірку концепції та мінімізацію ризиків на етапі розроблення програмного забезпечення.

**Цілі прототипування:**

**1) перевірка концепції та моделювання процесів** – прототип дозволяє максимально наблизити бачення майбутньої системи до реального функціонування, включаючи емуляцію робочих процесів з використанням тестових даних, а також оцінити зручність використання; робота з прототипом дозволяє своєчасно скоригувати вимоги до майбутнього ПЗ і передавати у розроблення лише перевірені та ретельно деталізовані завдання;

**2) керування інвестиціями та мінімізація ризиків** – верифікація концепції та деталізація вимог, яка досягається в процесі створення та оцінювання прототипу, дозволяє мінімізувати ризики при інвестуванні у розроблення ПЗ шляхом завчасного виявлення потенційних «вузьких» місць, точної пріоритезації задач та реалістичного планування бюджетів та термінів. Такий підхід забезпечує коректну реалізацію проектних вимог та ідей, закладених в концепцію майбутньої системи.

**Процес створення прототипу складається з таких кроків:**

1) визначення початкових вимог;

2) розроблення першого варіанту прототипу, який містить лише інтерфейс користувача системи;

3) вивчення прототипу замовником та кінцевими користувачами, одержання зворотного зв’язку про необхідні зміни та доповнення;

4) перероблення та покращення прототипу: з врахуванням одержаних зауважень та пропозицій змінюються як специфікації, так і прототип; після цього кроки 3 ф 4 можуть повторюватись.

**Два основних типи прототипування:**

**- швидке прототипування** (rapid або throwaway prototyping) – створюється макет, який на певному етапі буде залишений і не стане частиною готової системи. Основна перевага – у швидкості: у відповідь на свої вимоги замовник практично одразу одержує прототип інтерфейсу і одразу може уточнювати свої вимоги. Вартість зміни вимог на цьому етапі дуже низька, оскільки немає коду, який потрібно переписувати. Швидке прототипування виконується в найкоротші терміни, не обов’язково в межах тієї платформи та тих технологій, як і розроблювана система;

**- еволюційне прототипування** (evolutionary prototyping) – має за мету послідовно створювати макети системи, які будуть все ближче й ближче до реального продукту. Перевага такого підходу – на кожному кроці наявна робоча систем, яка нехай і не володіє всією необхідною функціональністю, але яка покращується з кожною ітерацією. При цьому немає витрат на код, який не буде використовуватись. Еволюційний підхід до прототипування обирається, якщо всі необхідні вимоги до моменту початку розроблення невідомі і будуть визначатись по мірі створення програми, тоді на кожному етапі реалізуються наявні та ясні вимоги. В деяких випадках, коли розробляється новітня система, аналогів якої немає, користувачі починають використовувати систему ще до того, як вона буде повністю дописана, адже система з неповною функціональністю краще, ніж її повна відсутність.

**Переваги застосування прототипування:**

- **зменшення часу, вартості, ризиків:** прототипування покращує якість специфікацій; чим пізніше проводяться зміни у специфікації, тим вони дорожчі, тому уточнення «чого ж замовники хочуть насправді» на ранніх стадіях розроблення знижує загальну вартість;

- **залучення користувача у процес розроблення:** прототипування залучає майбутніх користувачів до процесу розроблення і дозволяє їм бачити те, як саме виглядатиме майбутня програма, що дозволяє позбавитись від можливих розбіжностей в уявленні про програму між розробниками та користувачами.

**Недоліки застосування прототипування:**

**- недостатній аналіз:** концентрація зусиль на обмеженому прототипі може відволікати розробників від необхідного аналізу вимог на повну систему;

- **змішування прототипу та готової системи в уявленні користувачів:** користувачі можуть подумати, що прототип, який пропонується відкинути і є основою майбутньої системи, тому можуть вимагати від прототипу більш точної поведінки або розчаруватись у можливостях розробників;

- **надмірний час на створення прототипу:** ключова властивість прототипу – короткий час його створення; якщо ж розробники не приймають це до уваги, то вони витрачають час на створення надто складного прототипу та втрачають переваги від застосування прототипування взагалі.

Спірним є питання, чи застосовне прототипування взагалі, у тій чи іншій формі, до всіх типів проектів. Однак відомо, що найбільші переваги прототипування дає при розробленні систем, які мають розвинутий інтерфейс користувача. Інакше прототипування майже не дає реальних переваг.

**Варіанти використання прототипів:**

- як інструмент видобування, перевірки та затвердження вимог;

- як основу для написання специфікації вимог до ПЗ (SRS) та технічного завдання на етапі проектування;

- як техніку перевірки програмного дизайну на етапі проектування;

- як об’єкт дослідження юзабіліті-тестування (тестування на зручність використання);

- як зразок для розробників на етапі реалізації (конструювання);

- як зразок системи на етапі комерційної пропозиції;

- як зразок при тестуванні готового ПЗ;

- як зразок при прийманні-передачі роботи;

- як приклад рішення для демонстрації потенційним замовникам.

**Стандартизація розроблення ПЗ**

Стандартизація - це область, в якій постійно збільшується число учасників. Стандарти завжди грали особливу роль в телекомунікаційній галузі, оскільки вони є ключем до функціональної сумісності різних технологій: комп'ютерних систем, засобів зв'язку та передачі сигналів, програмного забезпечення та мультимедійних систем. У цій області основної технологічної проблемою є проблема порозуміння і взаємодії систем, джерелом якої служить використання виробниками таких технологій різного технологічного моделювання та стандартизації. Розробка технологій на основі загальних стандартів дозволить реалізувати принцип взаємної відкритості систем, який є найважливішою характеристикою інформаційного суспільства. Технологічно цей принцип вільного взаємодії та властивості відкритих систем називається *інтероперабельність,* також даний термін визначається і законодавчо. Інтероперабельність являє собою одне з головних властивостей відкритих систем і досягається за рахунок використання узгоджених наборів стандартів.

Розвиток будь-якої галузі економіки обов’язково супроводжується формалізацією використовуваних підходів та появою стандартів різного рівня. На ранніх етапах окремі підприємства формалізують внутрішні процеси, щоб забезпечити повторюваність результатів процесу або створення певного продукту. Для полегшення взаємодії підприємств та зручності споживачів розробляються ***галузеві стандарти.***

Розвиток кожного виду господарської діяльності приводить до потреби у державних засобах забезпечення якості продукції або процесу, тому розробляються та затверджуються ***державні стандарти***.

Для поліпшення умов співробітництва, розроблення загальнозрозумілих правил конкуренції на міжнародному ринку створюються об’єднання галузевих органів стандартизації, результатом діяльності яких є ***регіональні стандарти*** (діють у обмеженому переліку держав, які приєдналися) або ***міжнародні стандарти.***

**Стандарти на розроблення програмного забезпечення**

**На сьогодні діють такі стандарти, які регламентують процес розроблення ПЗ:**

* ***ГОСТ 34.601-90[[1]](#footnote-1)***– державний стандарт, що поширюється на автоматизовані системи і встановлює стадії та етапи їх створення. У стандарті міститься опис змісту робіт на кожному етапі.
* ***ISO/IEC 12207***. ***Systems and software engineering – Software Life Cycle Processes***– міжнародний стандарт на процеси розроблення та організацію життєвого циклу ПЗ. Поширюється на всі види замовленого ПЗ. Стандарт не містить опису фаз, стадій та етапів.
* ***Guide to the Software Engineering Body of Knowledge (SWEBOK)[[2]](#footnote-2)*** – Керівництво до зведення знань з програмної інженерії – галузевий стандарт Інституту інженерів з радіоелектроніки та електротехніки (IEEE), що систематизує основні види діяльності з програмної інженерії.

**Міжнародні стандарти ISO**

*ISO 12207* Systems and software engineering – Software Life Cycle Processes *- базовий стандарт* процесів життєвого циклу ПЗ, орієнтований на різні види ПЗ і типи проектів автоматизованих систем, в яких ПЗ є однією з складових частин. Стандарт визначає стратегію і загальний порядок в створенні та експлуатації ПЗ, він охоплює життєвий цикл від концептуалізації ідей до завершення проекту. Доцільність спільного використання стандартів на інформаційні системи і на ПЗ обумовлюється одним з положень ISO 12207, згідно з яким процеси, використовувані під час життєвого циклу ПЗ, повинні бути сумісні з процесами, використовуваними під час життєвого циклу автоматизованої системи.

Згідно ISO 12207, *система -* це об'єднання одного або декількох процесів, апаратних засобів, програмного забезпечення, обладнання та людей для забезпечення можливості задоволення певних потреб або цілей.

У стандарті ISO 12207 не передбачено будь-яких етапів (фаз або стадій) життєвого циклу інформаційної системи. Даний стандарт визначає лише ряд процесів, кожен процес, дія або завдання ініціюються і виконуються іншим процесом в міру необхідності, причому немає заздалегідь визначених послідовностей (природно, при збереженні логіки зв'язків по вихідним відомостям завдань і т. п.) .

В ISO 12207 усі процеси ЖЦ ПЗ розподілені на три групи

 Рис.1 Процеси ЖЦ ІС відповідно до стандарту ISO 12207

У стандарті ISO 12207 описані п'ять **основних процесів ЖЦ ПЗ**:

* *процес придбання* визначає дії підприємства-покупця, що здобуває інформаційну систему, програмний продукт або службу програмного забезпечення;
* *процес супроводу* визначає дії персоналу, який забезпечує супроводження програмного продукту, тобто управління модифікаціями програмного продукту, підтримку його поточного стану та функціональної придатності; сюди ж відносяться установка програмного виробу на обчислювальній системі і його видалення.
* *процес поставки* визначає дії підприємства-постачальника, яке постачає покупця системою, програмним продуктом або службою програмного забезпечення;
* *процес розробки* визначає дії підприємства-розробника, яке розробляє принцип побудови програмного виробу та програмний продукт;
* *процес функціонування* визначає дії підприємства-оператора, яке забезпечує обслуговування системи в цілому (а не тільки програмного забезпечення) в процесі її функціонування в інтересах користувачів. На відміну від дій, які визначаються розробником в. Інструкціях з експлуатації, визначаються дії оператора з консультування користувачів, отриманню зворотного зв'язку та ін., які він планує сам і бере на себе відповідні обов'язки;

    Крім основних, стандарт ISO 12207 обумовлює 8 допоміжних процесів, які є невід'ємною частиною всього життєвого циклу програмного виробу та забезпечують належну якість проекту програмного забезпечення.

***Допоміжні процеси*** призначені для підтримки виконання основних процесів, забезпечення якості проекту, організації верифікації та тестуванняПЗ. Це процеси: вирішення проблем; документування; управління конфігурацією; забезпечення якості; верифікації; атестації; спільної оцінки; аудиту.

***Організаційні процеси*** визначають дії та завдання замовників та розробників для керування процесами у ході проекту.Це процеси: управління; створення інфраструктури; удосконалення; навчання. Під процесом удосконалення розуміється не удосконалення інформаційної системи або програмного забезпечення, а поліпшення самих процесів придбання, розробки, забезпечення якості і т. д., реально здійснюваних в організації.  Процес адаптації визначає основні дії, необхідні для адаптації цього стандарту до умов конкретного проекту.

**Особливості стандарту** ISO **12207**

* Стандарт ISO 12207 має динамічний характер, обумовлений способом визначення послідовності виконання процесів і завдань, при якому один процес при необхідності викликає інший або його частину. Такий характер дозволяє реалізувати будь-яку модель життєвого циклу. Відповідно до стандарту ISO 12207, модель життєвого циклу - це структура, яка містить процеси, дії і завдання, які здійснюються в ході розробки, функціонування та супроводження програмного продукту протягом усього життя системи, від визначення вимог до завершення її використання.
* Стандарт ISO 12207 забезпечує максимальний ступінь адаптивності. Безліч процесів і задач сконструйовано так, що можлива їх адаптація у відповідності з конкретними проектами інформаційних систем. Ця адаптація зводиться до виключення процесів, видів діяльності і завдань, які не застосовані в конкретному проекті.  Згідно ISO 12207, додавання унікальних або специфічних процесів, дій і завдань має бути обумовлено в контракті між сторонами. Причому «контракт» розуміється в самому широкому сенсі - від юридично оформленого документа до неформального угоди. Ця угода може бути визначене навіть єдиною стороною - як завдання, поставлене самому собі.
* Стандарт принципово не містить опису конкретних методів дій, а тим більше - заготовок рішень або документації. Він лише описує архітектуру процесів ЖЦ ПЗ , але не конкретизує в деталях, як реалізовувати або виконувати послуги і завдання, включені в процеси. Даний стандарт не вказує імена, формати або точний зміст одержуваної документації. Рішення такого типу приймаються сторонами, що використовують стандарт.
* Забезпечення якості різними процесами виконується з різною передбаченої ступенем організаційної незалежності контролюючої діяльності аж до обов'язкових вимог до повної незалежності перевіряти персоналу від будь-якої прямої відповідальності. Щодо перевірки об'єктів, то контроль цього виду передбачений на самих ранніх кроках розробки, починаючи з аналізу системних вимог шляхом їх перевірок на відповідність потребам придбання.
* Ступінь обов'язковості розглянутого стандарту така: після рішення організації про застосування ISO 12207 в якості умови торгових відносин вступає її відповідальність за зазначення мінімального набору необхідних процесів і завдань, які забезпечують узгодженість з цим стандартом.
* Стандарт містить гранично мало описів, спрямованих на проектування бази даних, що виправдано тим, що різні системи і різні прикладні комплекси програмного забезпечення можуть не тільки використовувати вельми специфічні типи баз даних, але і взагалі не використовувати базу даних.

***Цінність стандарту ISO 12207***полягаєу тому, що він містить набори завдань, характеристик якості, критеріїв оцінки і т. п., що дають всебічне охоплення проектних ситуацій. Наприклад, при виконанні аналізу вимог до системи передбачається, що:

* розглядається область застосування системи для визначення вимог, пропонованих до системи;
* специфікація вимог системи повинна описувати функції і можливості системи, області застосування системи, організаційні вимоги і вимоги користувача, безпека, захищеність, людські фактори, ергономіку, зв'язку, операції та вимоги супроводу; проектні обмеження та кваліфікаційні вимоги.

    Далі, при виконанні аналізу вимог до програмного забезпечення передбачено 11 класів характеристик якості, які використовуються пізніше при забезпеченні якості. При цьому розробник повинен встановити і документувати у вигляді вимог до програмного забезпечення наступні специфікації і характеристики:

* функціональні та можливі специфікації, включаючи виконання, фізичні характеристики та умови середовища експлуатації, при яких одиниця програмного забезпечення повинна бути виконана;
* зовнішні зв'язки (інтерфейси) з одиницею програмного забезпечення;
* вимоги кваліфікації;
* специфікації надійності, включаючи специфікації, пов'язані з методами функціонування та супроводу, впливу навколишнього середовища та ймовірністю травми персоналу;
* специфікації захищеності, включаючи специфікації, пов'язані з компрометацією точності інформації;
* людські фактори специфікацій з інженерної психології (ергономіці), включаючи пов'язані з ручним керуванням, взаємодією людини і устаткування, обмеженнями на персонал та областями, потребуючими в концентрованому людському уваги, які є чутливими до помилок людини і навчанню;
* визначення даних і вимог до бази даних;
* установочні та приймальні вимоги поставляється програмного продукту в місцях функціонування та супроводу (експлуатації);
* документацію користувача;
* робота користувача і вимоги виконання;
* вимоги сервісу користувача.

     Відповідно до стандарту IS012207, вимога кваліфікації - це набір критеріїв або умов (кваліфікаційні вимоги), які повинні бути задоволені для того, щоб кваліфікувати програмний продукт як такий, що задовольняє умовам та його специфікаціям і готовий для використання в навколишньому середовищі.

     Хоча стандарт не вказує конкретної моделі життєвого циклу або методу розробки, він визначає, що сторони-учасники при використанні стандарту відповідальні за наступне:

* вибір моделі життєвого циклу для розроблювального проекту;
* адаптацію процесів і задач стандарту до цієї моделі;
* вибір та застосування методів розробки програмного забезпечення;
* виконання дій і завдань, придатних для проекту програмного забезпечення.

В стандарті ISO міститься 12207 найбільш широкий набір процесів, дій і завдань, що охоплює більшість можливих ситуацій при максимальній адаптованості. Він містить мінімум обмежень і конкретних рекомендацій. При використанні ISO 12207 детальні визначення процесів, форм документів тощо доцільно виносити в різні функціональні стандарти, відомчі нормативні документи або фірмові методики, які можуть бути використані або не використані в кожному конкретному проекті.

Для підтримки практичного використання стандарту ISO 12207 розроблені такі технологічні документи: Керівництво для ISO/IEC 12207 (ISO/IEC TR 24748-3:2011 Systems and software engineering - Life cycle management - Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes)) та Керівництво з використання ISO/IEC 12207 в керуванні проектами (ISO/IEC TR 16326:2009 Systems and software engineering - Life cycle processes - Project management).

***Контрольні запитання для самоперевірки***.

1. Що визначають моделі конструювання ПЗ та які складові вони включають?
2. Визначте основні стратегії конструювання ПЗ?
3. В чому полягає CASE-технологія?
4. Коротко охарактеризуйте основні моделі конструювання ПЗ.
5. Які переваги надає UML при конструюванні ПЗ?
6. Наведіть приклад засобів моделювання для конструювання ПЗ.
7. Визначте переваги та недоліки прототипування.

**Рекомендована література**

1. Лавріщева К.М. Програмна інженерія. Електронний підручник: http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf
2. Лавріщева К.М. Програмна інженерія. – Підручник.–К.:Академперіодика, 2008.–415с.
3. И. Соммервиль. Инженерия программного обеспечения,  
    6 изд. – И.д. "Вильямс", 2002.
4. 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>.
5. ДСТУ ISO/IEC 15288:2005. Інформаційні технології. Процеси життєвого циклу системи (ISO/IEC 15288:2002, IDT) - К.: Держстандарт України, 2005. - К.: Держстандарт України, 2004.
6. ДСТУ ISO/IEC/IEEE 12207:2018 Інформаційні технології. Процеси життєвого циклу програмного забезпечення - К.: Держстандарт України, 2018.

***Для самостійного вивчення***: Ознайомитися зі стандартом ISO 12207 та стандартами ЕСПД (ГОСТ 19). Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

1. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. – М., 1991. [↑](#footnote-ref-1)
2. <https://studfile.net/preview/3652697/> неофіційний переклад російськлю [↑](#footnote-ref-2)