**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](https://uk.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B0%D0%BC%D0%B8) 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:

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

q *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;

q *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 ТППЗ 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 крупномасштабних 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 зворотнього 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 замоники хотять 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.

***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.

***Для самостійного вивчення***: Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.