# **Лекція** з навчальної дисципліни: **«Крос-платформне програмування»**

## Тема 3. Методи створення компонентів

## Заняття 3.1. (ЛЗ) Маршалінг. Розробка компонентів, об’єктів і сервісів

О. Є. КОВАЛЕНКО,
к.т.н., доцент СК №5

**Навчальні питання:**

1. Маршаллінг.
2. Об’єкти та інтерфейси, що ними надаються.
3. Типові компоненти.

**Література**:


1. Кулямин В. В.. Технологии программирования. Компонентный подход. М. Интернет-университет информационных технологий — БИНОМ. Лаборатория знаний, 2007. – 316 с. — Режим доступу: http://panda.ispras.ru/~kuliamin/lectures-sdt/sdt-book-2006.pdf.
2. Степанов Е.О. Кросс-платформенные и многозвенные технологи. / Интернет-университет информационных технологий - ИНТУИТ.ру, 2010 – Режим доступа: http://www.intuit.ru/department/se/crosspl/
3. Лавріщева К.М. Програмна інженерія. – К.: Академперіодика, 2008. – 319 с.

# 1. Маршаллінг. Перетворення форматів даних

**Маршаллінг даних** – перетворення до формату даних приймаючої серверної платформи, з урахуванням порядку і стратегії вирівнювання, прийнятої на цій платформі, даних програм, розташованих на різних типах комп'ютерів, які передають ці дані один одному з використанням стандартних протоколів.

**Демаршаллінг даних** – це зворотне перетворення даних (тобто отриманого результату) до виду клієнтської передавальної програми.

# Стандарт ISO/IEC 11404–96 (ГОСТ 30664-99 ) з незалежних від мов типів даних

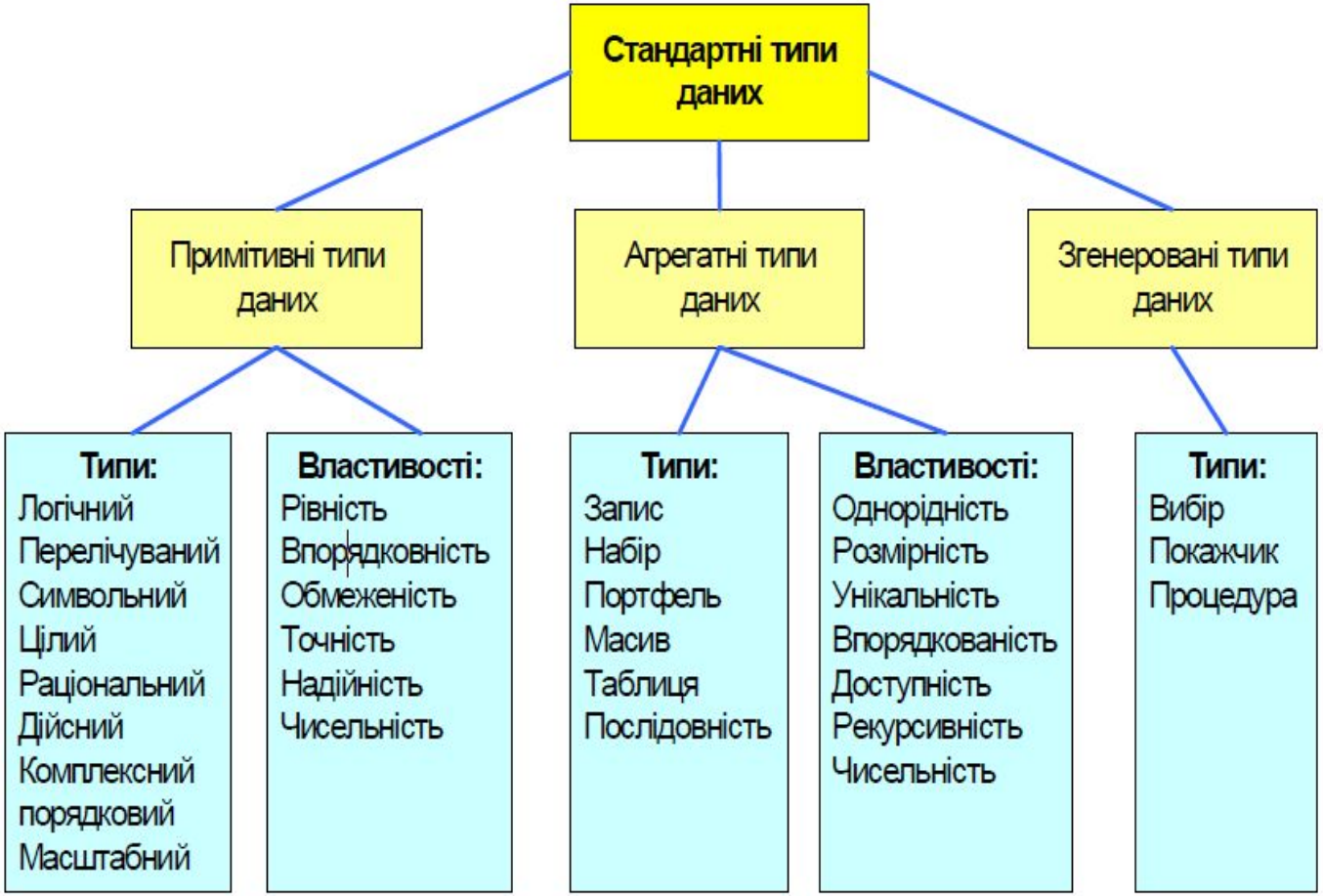



Рис. 3.1. Незалежні від МП типи даних стандарту ISO/IEC 11404–1996

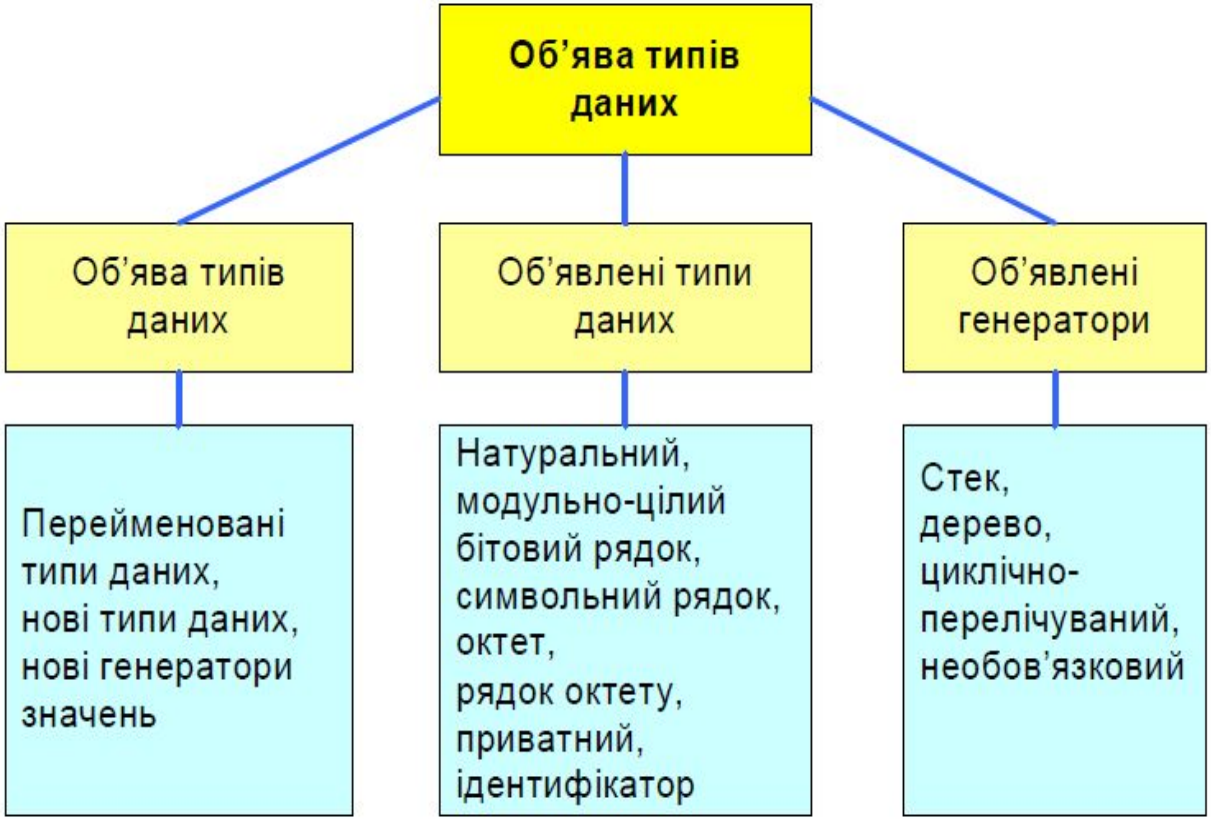



Рис. 3.2. Об'ява типів даних у стандарті ISO/IEC 11404–1996

LI (Language Independent)-мова стандарту рекомендує такі види перетворення даних:

- – зовнішнє перетворення типів даних МП в LI-типи даних;
- – внутрішнє перетворення з LI-типу даних в тип даних МП;
- – зворотне внутрішнє перетворення.

Зовнішнє перетворення типів даних і генераторів типів даних полягає в такому:

- перетворення кожного примітивного типу з зовнішнього типу даних зв'язується з одним LI-типом даних;
- перетворення кожного внутрішнього типу даних перетворення визначає зв'язок між припустимим значенням внутрішнього типу даних і еквівалентним значенням відповідного LI-типу даних;
- для кожного значення LI-типу даних, що бере участь в перетворенні, визначається існуванням значення будь-якого внутрішнього типу даних, що перетворюється в LI-тип даних з узяттям цього значення.

# Засоби перетворення даних і форматів

– стандарти кодування даних (XDR – eXternal Data Representation, CDR – Common Representation Data), NDR – Net Data Representation) і методи їхнього перетворення;

– МП і механізми звернення компонентів один до одного;

– мови опису інтерфейсів компонентів – RPC, IDL і RMI для передачі даних між різними компонентами.

# XDR

У XDR-стандарті цілі числа з порядком «від молодшого» зводяться до порядку байтів «від старшого» і назад. Перетворення даних – це кодування (code) або декодування (decode) XDR-процедурами форматування простих і складних типів даних. Кодування – це перетворення з локального уявлення в XDR-уявлення і запис в XDR-блок. Декодування – це читання даних з XDR-блоку і перетворення в локальне уявлення заданої платформи.

Вирівнювання даних – це розміщення значень базових типів з адреси, кратної дійсному розміру в байтах (2, 4, 8, 16). Межі даних вирівнюються за найбільшою довжиною (наприклад, 16).

# CDR

CDR-стандарт середовища CORBA забезпечує перетворення даних у формати платформи, що їх передає або приймає. Маршаллинг даних виконує інтерпретатор TypeCode і брокер ORB.

Процедури перетворення складних типів вміщують:

– додаткові коди для представлення цілих чисел і чисел з плаваючою точкою (стандарт ANSI/IEEE);

– схему вирівнювання значень базових типів в середовищі компілятора;

– базові типи (signed і unsigned) в IDL, а також плаваючому типі подвійної точності та ін.

# XML

XML-стандарт забезпечує усунення неоднорідності у взаємозв'язках компонентів у різних МП за допомогою XML-формату даних, який враховує різні платформ і середовища.

Проміжні середовища (CORBA, DCOM, JAVA та ін.) мають у своєму складі спеціальні функції, аналогічні XML – альтернатива сервісам CORBA в плані забезпечення взаємозв'язків різномовних програм.

XML має різну системну підтримку: браузер Internet Explorer для візуалізації XML-документів, об'єктна модель DOM (Document Object Model) для відображення XML-документів і інтерфейс IDL в системі CORBA.

# Перетворення даних з баз даних

Перетворення даних БД пов'язане з різницею логічних структур даних, а також з такими проблемами:

1) багатомодельність представлення даних (ієрархічні, мережні, реляційні) в різних БД і СКБД;

2) різниця в логічних структурах даних, в довідниках, класифікаторах і в системах кодування інформації;

3) використання різних мов для представлення текстової інформації;

4) різні типи СКБД і постійний розвиток даних БД в процесі експлуатації.

# Процес перетворення і формування нової БД із старих БД

# 2. Об’єкти та інтерфейси, що ними надаються

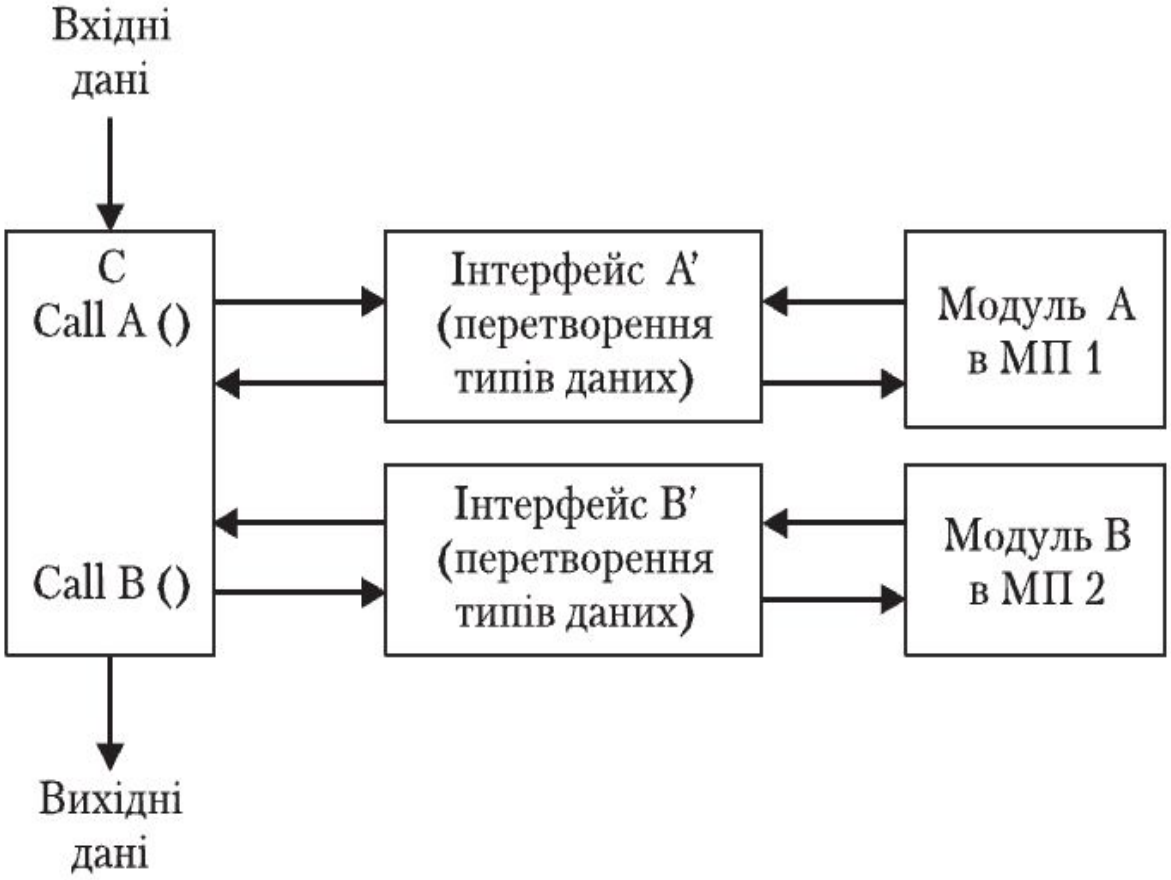
Види інтерфейсів: мовні, програмні, апаратні, призначені для користувача, цифрові і т.п.

Програмний (API) і/або апаратний інтерфейс (port) – це способи перетворення вхідних/вихідних даних під час об'єднання комп'ютера з периферійним обладнанням.

У МП – це програма або частина програми, в якій визначаються константи, змінні, параметри і структури даних для передачі іншим.

У програмуванні термін інтерфейс означає набір операцій, що забезпечують визначення видів послуг і способів їхнього отримання від програмного об'єкта, що надає ці послуги.

# Схема зв’язків модулів A і B із C через інтерфейси A’ і B’



# Структура представлення класу

| **Клас** | |
|---|---|
| **Зовнішнє подання** | **Внутрішнє подання** |
| Інтерфейсні операції: | Реалізація операцій класу |
| – публічні, доступні всім клієнтам; | визначення поведінки |
| – захищені, доступні класу і підкласу; | |
| – приватні, доступні класу | |

# Парадигма об'єктів

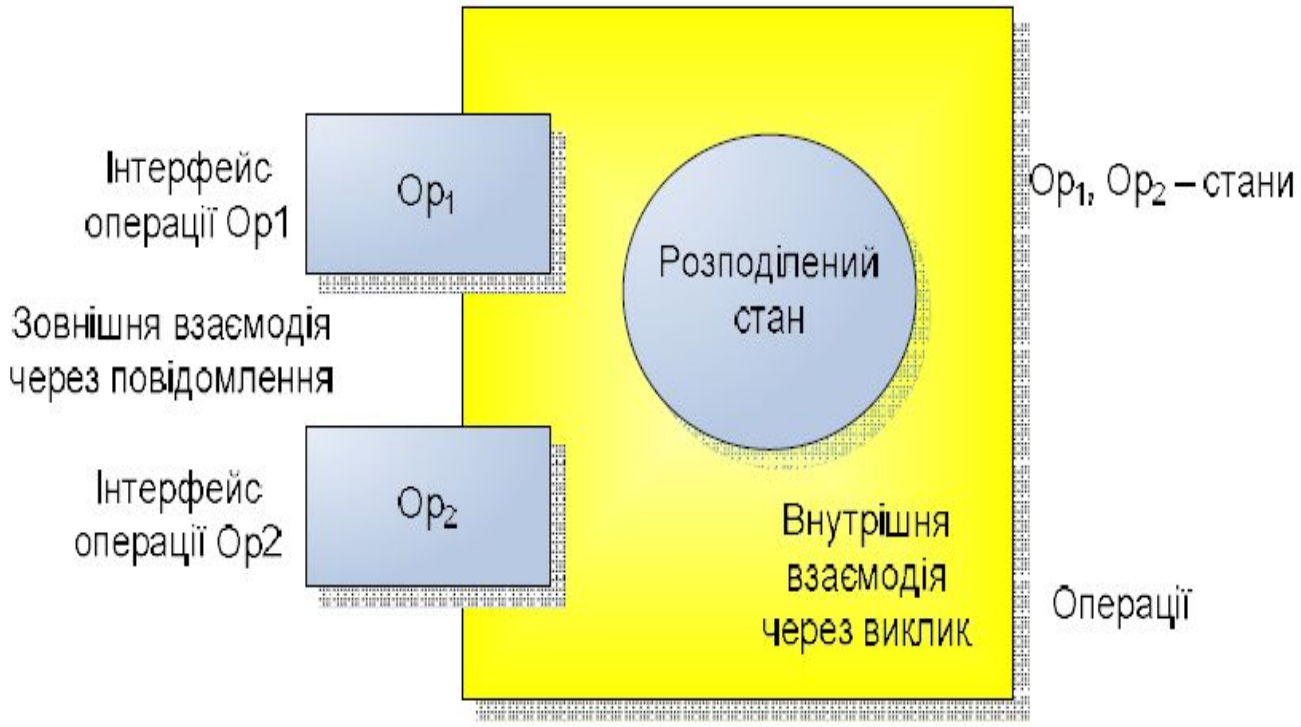
Сутьпарадигми переходу від алгоритмів обчислень до взаємодії об'єктів полягає в тому, що **обчислення** і **взаємодія об'єктів** розглядалися як дві ортогональні концепції.

**Взаємодія** – це деяка дія (action), але не обчислення,

а **повідомлення** – не алгоритм, а дія, відповідь на яку залежить від послідовності операцій (Op), що впливають на стан розподіленої (shared state) пам'яті локальної програми.

Операції інтерфейсу (Op1 і Op2) належать до класу неалгоритмічних і забезпечують взаємодію об'єктів через повідомлення.

# Інтерфейс взаємодії через операції інтерфейсу (за Вегнером)



Вегнер розглядає модель взаємодії як **узагальнення машини Т'юрінга** – розподіленої інтерактивної моделі взаємодії об'єктів з вхідними (**input**) і вихідними (**output**) діями і можливістю просування в ній потенційно нескінченного вхідного потоку (**запитів, пакетів**) в заданому інтервалі часу.

Відкриті системи надають будь-яким застосуванням різного роду послуги: керування віддаленими об'єктами, обслуговування черг і запитів, обробка інтерфейсів і т.п.

Доступ до послуг здійснюється за допомогою механізмів:

– виклику віддалених процедур **RPC** (Remote Procedure Call) в системах ONC SUN, OSF DSE;

– скріплення розподілених об'єктів і документів в системі **DCOM**;

– мови опису інтерфейсу **IDL** (Interface Definition Language) з підтримкою його брокером – **ORB** (Object Request Broker) в системі **CORBA**;

– виклику **RMI** (Remote Methods Invocation) в системі **JAVA** та ін.

# 3. Типові компоненти

**Інженерія КПВ** – це систематична і цілеспрямована діяльність з підбору реалізованих і представлених у вигляді КПВ програмних артефактів, аналізу їхніх функцій для додавання їх у проектовану систему як готових компонентів та інтеграції з іншими компонентами.

# Процеси КПВ у ЖЦ

***Перший процес* – це побудова КПВ** шляхом:

- – вивчення спектра розв’язуваних задач ПрО, виявлення серед них загальних властивостей і функцій;
- – опис компонентів, реалізуючих виявлені функції у вигляді звичайних компонентів або КПВ;
- – зберігання виготовлених компонентів у каталозі і організація пошуку необхідних компонентів з запитів користувачів.

***Другий процес* – конструювання нових систем з готових компонентів** шляхом:

- – визначення цілей майбутній системи і вимог, що висуваються до неї;
- – пошуку в каталозі готових компонентів, які можуть відповідати вимогам до нової системи;
- – зіставлення окремих цілій нової розробки з можливостями знайдених КПВ і прийняття рішень про доцільність і місце їхнього застосування в системі;
- – інтеграція КПВ у нову розробку із забезпеченням інтерфейсу з підсистемами та іншими компонентами.

# Критерії успіху використання КПВ:

1) повторне використання готових компонентів вимагає менших трудовитрат, ніж їхня нова розробка;
2) пошук придатних для використання компонентів вимагає менших зусиль, ніж реалізація необхідних функцій у проектованій системі;
3) розгортання компонентів в нових умовах середовища, потребує менших трудовитрат, ніж знов виконаної розробки.

# Артефакти

Під *артефактом* розуміється реальна порція інформації, яка може створюватися, змінюватися і використовуватися при виконанні діяльності, пов'язаної з розробкою ПС різного призначення.

Артефактами можуть бути:

– моделі ПрО в термінах понять і лексики деякої предметної області;

– готові компоненти ПС, КВП або окремі частини системи;

– проміжні продукти процесу розроблення ПС (вимоги, постановки завдань, архітектура та ін.);

– описи процесу проектування ПС (специфікація, модель, каркас і т.п.)

– діаграми, патерни і т.п.

# Модель ЖЦ КПВ

– аналіз об'єктів і зв’язків у ПрО, яка реалізується, для виявлення КПВ, що мають загальні властивості, притаманні групам об'єктів цієї області;

– адаптація наявних в базі репозитарію КПВ, розроблення нових функціональних компонентів, не представлених у цій базі, і доведення їх до рівня повторного використання;

– розроблення інтерфейсів компонентів і розміщення їх в депозитарії інтерфейсів системи;

– інтеграція КПВ з їхніми інтерфейсами та іншими елементами створюваної системи і формування конфігурації цієї системи.

# Прикладні і загальносистемні КПВ

*Прикладні компоненти* виконують окремі задачі і функції прикладної області (домени бізнесу, комерції, економіки і т.п.), які можуть використовуватися надалі як прикладні системи в інших доменах з аналогічними функціями.

До *загальносистемних компонентів* належать компоненти загального і універсального призначення, а також загальносистемні сервісні засоби, які забезпечують системне обслуговування і надають різні види сервісу для багатьох створюваних програмних систем різного призначення.

# Види КПВ

- – процедури і функції на МП високого рівня;
- – алгоритми, програми;
- – класи об'єктів і абстрактні класи;
- – структури даних і часто використовувана інформація (наприклад, інформаційні ресурси Інтернету);
- – API, IDL-модулі в бібліотеках (наприклад, GUI, графіка і ін.);
- – веб-ресурси і сервіси;
- – засоби розгортки систем і компонентів в операційному середовищі (наприклад, CORBA, COM, .NET);
- – готові розв’язки у вигляді абстракцій – патерни, фрейми та ін.

# Специфікація КПВ

Модель специфікації компонента має такий вигляд:

**КПВ = (*T*, *I*, *F*, *R*, *S*)**,

де

*T* – тип компонента;

*I* – множина інтерфейсів компонента;

*F* – функціональність компонента;

*R* – реалізація, прихована частина – програмний код;

*S* – сервіс для взаємодії з середовищем або набір правил розгортки.