ДИАГРАММЫ КЛАССОВ ООП: ФОРМАЛИЗАЦИЯ И АНАЛИЗ

Дмитрий Буй[1], Сергей Компан[2]

Киевский национальный университет имени Тараса Шевченка, Украина
[1]buy@unicyb.kiev.ua, [2]robin_2005@mail.ru

Аннотация В статье приводится короткий сравнительный анализ работ, посвящённых формальным моделям объектно-ориентированного программирования (ООП). Предмет исследования – модель диаграммы классов (соответствующее частично упорядоченное множество). Модель класса (спецификация класса) —- пара бинарных функциональных отношений, одна компонента уточняет атрибуты, вторая — методы. Наследование уточняется как включение графиков функций. Над классами вводятся две операции –– пересечение и сочленение. Формальные результаты: структура частично упорядоченного множества классов, свойства операции наложения, в терминах которой вводится операция сочленения: идемпотентность, ассоциативность, критерий коммутативности. Модель диаграммы классов можно использовать для анализа структуры классов, который может помочь для выделения подсистем клонов и оптимизации самой диаграммы.

Ключевые слова: формальные модели ООП, диаграммы классов, спецификация класса, операции над классами, клон

1. Вступление

Основы объектно-ориентированного подхода (ООП) были заложены в конце 60-х годов. В основе ООП лежит объектная модель данных (ОМД). Появление ОМД связано с появлением абстрактных типов данных. В 1971 году Парнас (D. L. Parnas) в работе [1] предложил идею инкапсуляции (сокрытия информации). В 80-х годах XX столетия Г. Буч (G. Booch) в своей книге [2] предложил использовать идеи ООП для разработки программных продуктов промышленного уровня. Благодаря ООП появилась потенциальная возможность выхода из кризиса, связанного с проектированием и реализацией сложных программных систем (ПС). Таким образом, ООП стало очередной ступенью к созданию методов разработки надежных ПС. В результате ускорился процесс создания программных комплексов, в основе которого лежат принципы ООП, а так же качественно снизился уровень сложности разработки ПС.

Согласно ООП предметная область (ПрО) представляется как множество объектов со свойствами, которые необходимы для определения и идентификации объектов, а также описаниями их поведения в рамках выбранной системы понятий на зафиксированном уровне абстракции. Предметная область,

которая моделируется объектами, сама является объектом и поэтому может быть отдельным объектом в составе другой ПрО. При моделировании объект ПрО получает хотя бы одно свойство, необходимое для его идентификации в множестве объектов ПрО [3]. В основе ООП лежат: абстракция, инкапсуляция, наследование, полиморфизм. Язык программирования, в котором реализованы эти понятия, является объектно-ориентированным и именно они придают языку гибкость при программной реализации сущностей ПрО.

2. Краткий анализ современного состояния ООП

Сразу необходимо заметить, что в связи с популярностью ООП как средства проектирования и разработки ПС существует достаточное количество соответствующей литературы [2–11]. Следует отметить ряд работ заведующей отделом Института программных систем НАН Украины, профессора Лаврищевой Е.М. [3,10,12], в которых уделяется большое внимание проектированию и программированию систем, в основе которых лежит ООП. Автор предлагает выделять четыре уровня абстрактного представления объектной модели (ОМ): обобщающий, структурно-упорядоченный, характеристический и поведенческий. На каждом из этих уровней применяется свой математический аппарат [10].

Отметим, что существует группа OMG (Object Management Group), в основе деятельности которой лежит пропагандирование и стандартизация ООП. Эта группа описала набор стандартов OMA (Object Management Architecture). Самым известным и широко применяемым на сегодняшний день является CORBA (Common Object Request Broker Architecture) —- спецификация взаимодействия разнотипных объектов в распределенной системе [13]. В данной модели для описания понятий применяется декларативный язык определения интерфейсов IDL (Interface Definition Language).

Для полной уверенности в том, что информационная система, построенная на основе ООП, будет удовлетворять исходным спецификациям и стабильно работать (будет гарантоустойчивой по терминологии [14]), необходимо выделить компоненты системы, формально их описать и верифицировать. В результате возникла необходимость формального определения понятий объекта, класса, операций над объектами и т.д. Для ООП построены формальные модели. Например, в работе [15] формально даётся определение основных понятий ООП: класса, объекта, наследования, инкапсуляции, полиморфизма с использованием математического аппарата теории множеств, функций, абстрактных автоматов (в этой же работе приводится обширная библиография по ООП). В работе [16] авторы строят формальную модель для объектных баз данных (БД), в основе которой лежит теория категорий. В [17,18] даны формальное определение основных понятий ООП, в основе определения которых лежит композиционное программирование, созданное академиком НАН Украины В. Н. Редько [19–21].

Особое внимание следует уделить вопросу построения объектной модели для объектных БД, так как в основе более-менее сложной информационной

системы лежит именно БД. Широкое распространение ООП резко обозначило проблему семантического разрыва между моделью языка программирования и моделью представления данных в БД. Объектная модель позволяет оперировать сложными структурами данных. В результате системы управления базами данных (СУБД) стали развиваться в трех направлениях. Первое — это построение отображения объектов в традиционную реляционную модель данных. Второе — включение понятий ООП в реляционные СУБД путем расширения типов данных, используемых в БД. Третье направление -— создание принципиально новых, объектно-ориентированных СУБД, которые в своей работе следуют стандарту ODMG [22].

В [23] авторы формально строят модель данных, в которую входят: конечное множество основных типов D, счетное множество объектных идентификаторов O, множество имен атрибутов A, множество имен методов M, множество имен классов C. Так как объект может содержать в себе данные, то различают примитивные значения (atomic values) и сложные значения (structured values). После этого авторы приводят синтаксическое описание класса (неформальное). Даётся формальное определение метода, который является функцией, отображающей сигнатуру в тело функции. Сигнатура формально описывается следующим образом — $c : m(\tau_1, \tau_2, \ldots, \tau_n) \to \tau_r$, где c -– имя класса, которому принадлежит метод, m —- имя метода, $\tau_1, \tau_2, \ldots, \tau_n$ — список типов параметров и τ_r — тип возвращаемого значения. Также в статье упоминается простое наследование, которое формально не определено. Приводится формальное определение объекта, который задаётся тройкой (o, ν, c), где $o \in O$ —- идентификатор объекта,v — значение объекта, $c \in C$ — класс, которому принадлежит объект. Также приводится формальное определение объектной БД. БД содержит схему (Schema) и экземпляры схемы (Instances). Схема описывается тройкой (C, σ, G), где C — как и ранее множество имен классов БД, σ — функция, которая сопоставляет именам классов собственно классы, G — множество глобальных переменных классов. Множества C и G выступают в качестве точки входа в БД.

Экземпляры схемы БД содержат конечное множество объектов и четыре функции: π_d — назначает объектам уникальные объектные идентификаторы (oid), π -— функция, которая по объекту определяет идентификатор oid, присвоенный объекту функцией π_d, υ — функция, которая сопоставляет объектным идентификаторам значения всех находящихся в БД объектов, γ -— функция, которая присваивает значения глобальным переменным объектов.

Вкратце обсудим построение объектной алгебры. По аналогии с реляционной моделью данных, в основе которой лежит реляционная алгебра Кодда, для ООП также рассматривают так называемую объектную алгебру, носителем которой является множество объектов [23–25]. Различаются эти алгебры только сигнатурными операциями над объектами. В [26] авторы вводят операции пересечения и сочленения (в работе она называлась объединением) над классами. В результате предлагается рассматривать вместо объектной алгебры алгебраическую систему. Формально ее можно описать как $< O, K; \Omega_{obj}; \Omega_{spec}, \leq >$, где O -— множество объектов классов, K -—

множество классов (спецификаций классов), Ω_{obj} — множество операций над объектами, Ω_{spec} -— множество операций над классами, а отношение $\leq \subseteq K \times K$ —- отношение частичного порядка, которое уточняет отношение наследования классов.

3. Постановка задачи

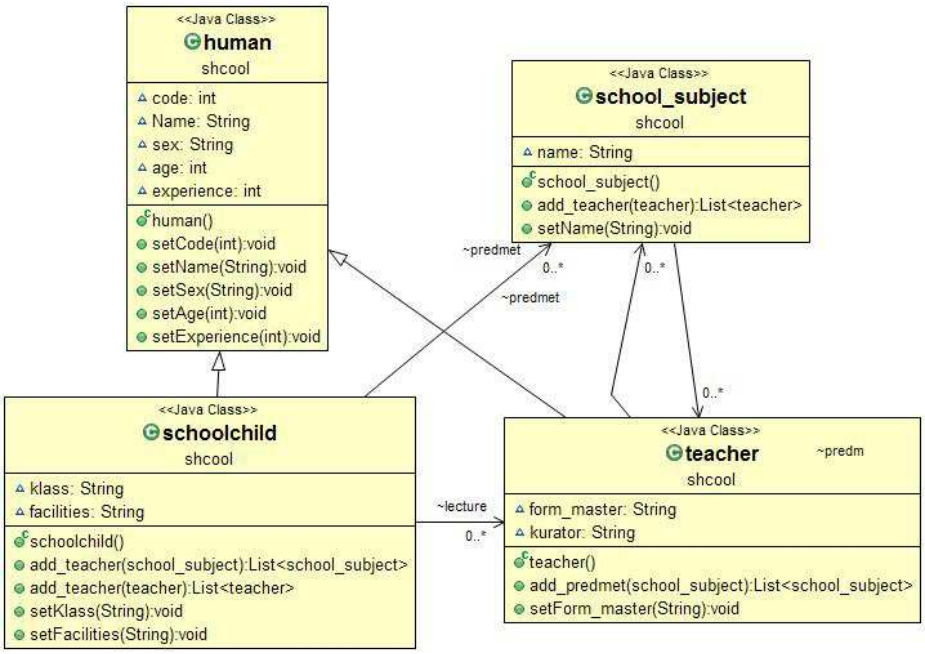

Рис. 1. Диаграмма классов ПрО “Школа”

Построение новой сложной системы, как правило, сопряжено с нетривиальными усилиями. В результате для построения сложной системы можно пойти двумя путями. Первый предполагает реализацию системы с нуля. При таком подходе одним из критериев успешного выполнения проекта является время реализации. Как правило, ограничение авторского коллектива во времени приводит к тому, что для проверки правильности программ коллектив выбирает тестирование вместо построения формальной модели и доказательства правильности работы такой формальной модели. Второй путь предполагает изучение уже реализованных подобных систем, с целью переноса работоспособной части функционала в новую систему. При таком подходе, если он целесообразный, мы можем эффективно использовать уже работающие компоненты других систем, при этом мы можем столкнуться с ситуацией,

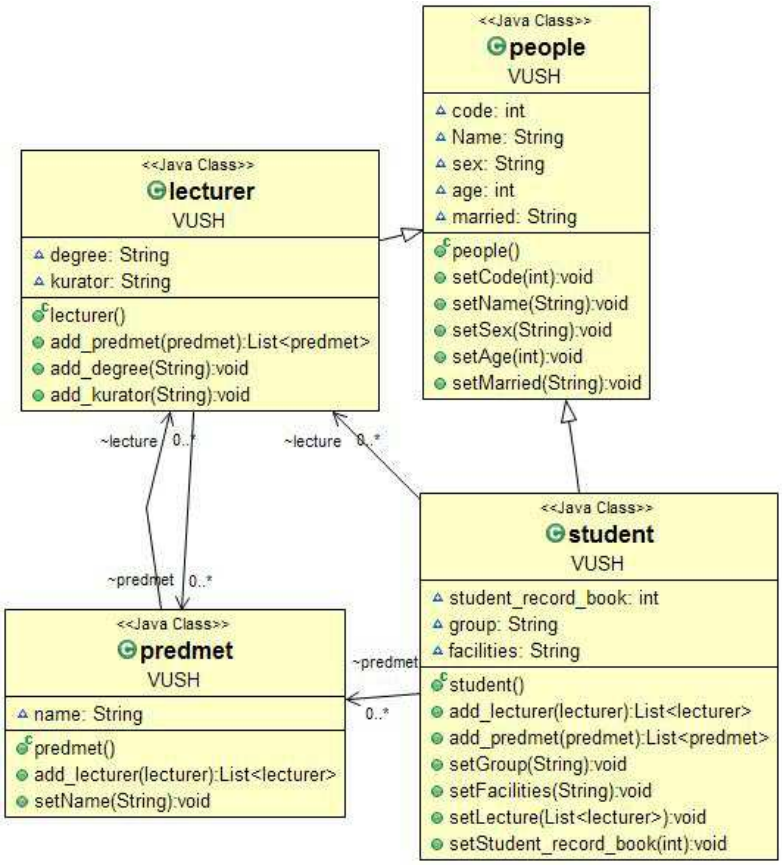

Рис. 2. Диаграмма классов ПрО “ВУЗ”

при которой в новую систему могут попасть “одинаковые” (в том или ином смысле) части кода (атрибуты и методы) — т.н. клоны. Для уменьшения избыточности кода (устранения клонов) в программе авторы статьи предлагают выносить общие части программы (атрибуты и методы) в новые классы, совокупность которых будет выступать аналогом FrameWork.

При моделировании ПрО авторы не берут смелость утверждать, что ПрО смоделирована в полной мере. Главная цель данного моделирования — показать идею использования операций над классами.

Пусть необходимо написать программу “Учебное заведение”, которая может использоваться как в школе, так и в ВУЗе. К примеру, пусть проведённый анализ созданного ПО по данной ПрО показал, что существуют готовые решения для школы (рис. 1) и для ВУЗа (рис. 2), но которые реализовывают только часть нашей ПрО. Для реализации нашей задачи можно объединить (сочленить) функциональные части исходных программ в одну. В результа-

те получим новую диаграмму классов (рис. 3), относительно которой сделаем несколько замечаний. Как указывается в обзоре [27] клоны могут быть удале-

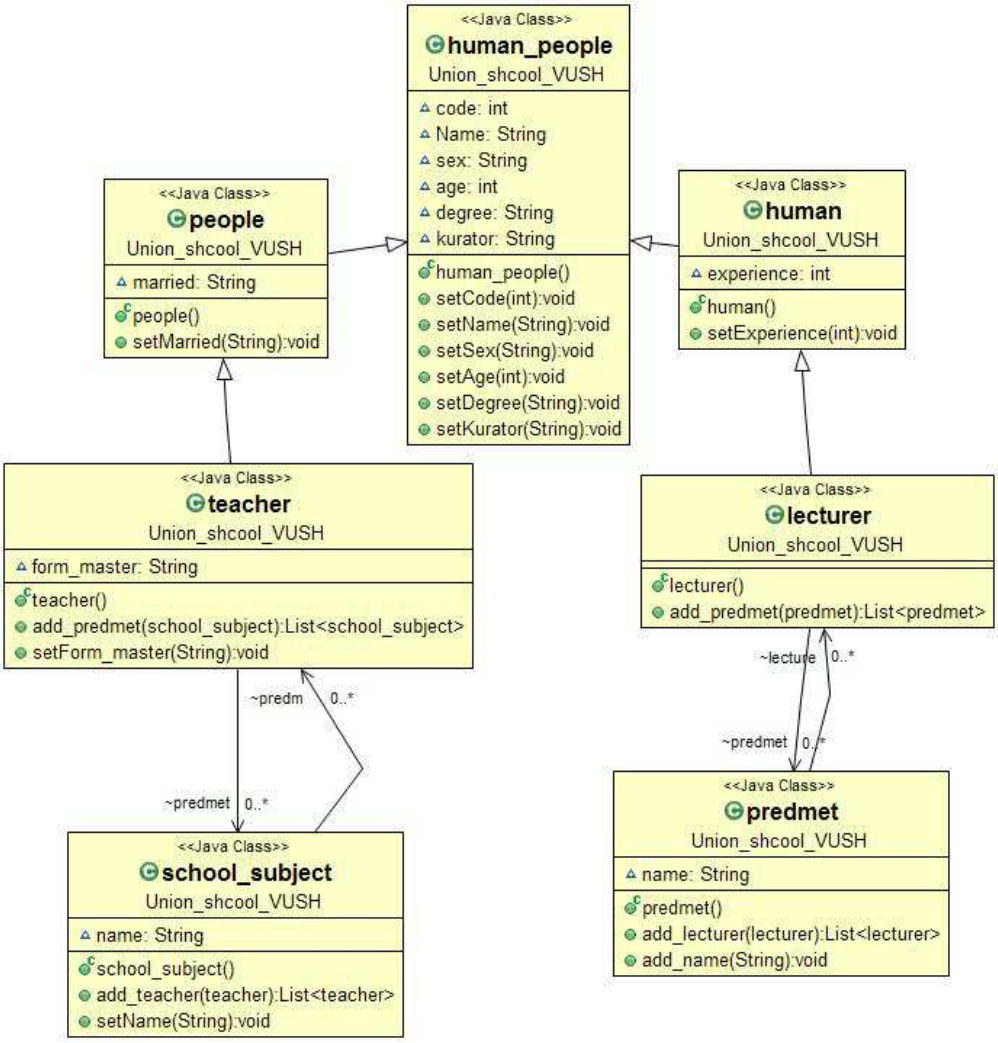

Рис. 3. Диаграмма классов ПрО “Школа и ВУЗ”

ны с использованием операций “выделение метода” (Extract Method) и “Подъем метода” (Pull Up Method), изложенных в монографии [28]. Выделение метода (в русскоязычной литературе называемое процедурной абстракцией) заключается в выделении общего кода в новую функцию (или метод класса, если речь об ООП). Подъем метода заключается в перемещении общего мето-

да двух классов в их базовый класс. Если два класса не являются потомками одного базового класса, то перед операцией Подъема метода можно создать новый базовый класс при помощи операции выделения родительского класса (Extract Superclass) [29]. В нашем подходе могут быть также выделены и общие атрибуты. Фактически своими преобразованиями над спецификациями классов мы изменяем внутреннюю структуру программ, не затрагивая их поведения и в какой-то степени улучшаем понимание кода. Целью работы является разработка теории ООП при минимальных предположениях о этой ПрО. Мы разделяем атрибуты и методы и рассматриваем функции, которые присваивают атрибутам и методам семантику (пока не уточняя какую именно семантику; это будет сделано при последующей конкретизации). Мы надеемся, что формальные результаты, представленные в следующем разделе, обосновывают право на существование выбранного нами уровня абстракции. По сути, в работе реализован принцип "разделяй и властвуй".

4. Математические результаты

Приведем формальное уточнение класса (синоним — спецификации класса). Под классом будем понимать пару $< s, \mu >$, где s —- функциональное бинарное отношение, которое атрибуту ставит в соответствие его тип (множество значений из универсального домена D), а μ —- функциональное бинарное отношение (функция), которое сигнатуре метода ставит в соответствие его реализацию (семантику, логику). Можно рассматривать μ как функциональное бинарное отношение, которое сигнатуре метода ставит в соответствие его тип; в этом случае получим определение интерфейса.[1]

Отношение наследования $\leq$ между классами уточняется так:
$< s, \mu > \leq < s', \mu' >$, если $s \subseteq s'$ и $\mu \subseteq \mu'$.

Приведем несколько определений, которые понадобятся в дальнейшем при изложении материала.

Определение 1. *Пусть α,β — функциональные бинарные отношения, тогда операция наложения определяется так: $\alpha \nabla \beta = \beta \bigcup \alpha \mid (dom\alpha \setminus dom\beta)$, где $dom\alpha$ и $dom\beta$ области определения соответственно функций α и β, а $\gamma \mid X$ — ограничение функции γ по множеству X, т.е. $\gamma \mid X = \gamma \bigcap (X \times \pi_2^2 \gamma)$ ($\pi_2^2 \gamma$ — проекция отношения γ по второй компоненте).*

Наложение иллюстрируется на рис. 4.

Уточним бинарную операцию сочленения $\coprod$ и пересечения $\cap$ классов. Обозначим $\mathbb{K}$ — множество классов, $K \in \mathbb{K}$ — класс. Операция сочленения является операцией вида: $\coprod : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$, где $< s_1, \mu_1 > \coprod < s_2, \mu_2 > =$
$= < s_1 \nabla s_2, \mu_1 \nabla \mu_2 >$.

[1] Несмотря на то, что тип метода входит в сигнатуру, нам для построения функционального бинарного отношения необходимо разделять (считать разными) методы, сигнатура которых отличается только типом возвращаемого значения. Такой подход необходим для исследования интерфейсов.

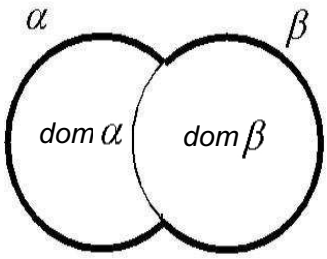

Рис. 4. Графическая интерпретация операции наложения $\alpha \triangledown \beta$ (β имеет приоритет

Операция сочленения классов $\coprod$ уточняет множественное наследование классов. При сочленении классов K_1 и K_2 получим новый класс K_3, для которого классы K_1 и K_2 будут базовыми (родительскими) классами, а класс K_3 будет играть роль производного. В производном классе K_3 классы K_1 и K_2 являются дополнением (расширением) одного к другому. Рассмотрим важнейшие частные случаи:

1) $doms_1 \bigcap doms_2 = \varnothing$ и $dom\mu_1 \bigcap dom\mu_2 = \varnothing$ — т.е. в классах-аргументах нет одинаковых атрибутов и методов. Тогда получим производный класс вида $K_3 =< s_1 \bigcup s_2, \mu_1 \bigcup \mu_2 >$;
2) $doms_1 \bigcap doms_2 \neq \varnothing$ или (и) $dom\mu_1 \bigcap dom\mu_2 \neq \varnothing$. Тогда возникает конфликт имен и нужно определить по определённому критерию (правилу), какой именно атрибут (метод) необходимо добавлять в производный класс; конфликт разрешается с помощью операции наложения.

Операция пересечения классов является операцией вида:
$\cap : \mathbb{K} \times \mathbb{K} \to \mathbb{K}$, где $< s_1, \mu_1 > \cap < s_2, \mu_2 >=< s_1 \bigcap s_2, \mu_1 \bigcap \mu_2 >$, здесь $\bigcap$ — обычное теоретико-множественное пересечение. При пересечении классов K_1 и K_2 получим новый класс K_3, для которого классы K_1 и K_2 будут производными, а класс K_3 будет играть роль базового (суперкласса). Классы K_1 и K_2 являются дополнением (расширением) суперкласса.

Сделаем несколько замечаний по поводу введенного определения класса. Если первая компонента класса $< s, \mu >$ пуста (т.е. $s = \varepsilon$ — пустое бинарное отношение), то получим определение библиотеки. Если же проекция по второй компоненте отношения μ (т.е. $\pi_2^2 \mu$) является множеством множеств, которые интерпретируется как области значений функций-семантик методов, то получим определение интерфейса.

Определение 2. *Бинарное отношение совместности $\approx$ в классе функциональных бинарных отношений вводится так: $\alpha \approx \beta \Leftrightarrow \alpha \mid X = \beta \mid X$, где $X = dom\alpha \bigcap dom\beta$.*

Содержательная интерпретация: функциональные отношения совместны, если они ведут себя одинаково на общих аргументах.

При доказательстве свойств операций над классами и установлении структуры семейства классов будут использоваться следующие абстрактные свойства ограничения [30]. Ниже $U_1, U_2, \ldots$ — произвольные бинарные отношения, $X, X_1, \ldots$ — произвольные множества, $\alpha, \beta, \ldots$ — как и ранее, произвольные бинарные функциональные отношения. Параметрический оператор $U \mapsto U \mid X$ обозначен через $\uparrow X$, $\circ$ — традиционная композиция бинарных отношений, $\pi_i^2\alpha$ — проекция бинарного отношения по i-той компоненте.

Предложение 1 (свойства ограничения). *Ограничение имеет свойства:*

1) $U_1 \subseteq U_2 \wedge X_1 \subseteq X_2 \Rightarrow U_1 \mid X_1 \subseteq U_2 \mid X_2$ *(монотонность ограничения, монотонность оператора* $\uparrow X$*);*
2) $\pi_1^2(U \mid X) = \pi_1^2 U \bigcap X$ *(проекция ограничения);*
3) $U \mid X = \varepsilon \Leftrightarrow \pi_1^2 U \bigcap X = \varnothing$ *(критерий пустоты ограничения); в частности,* $\varepsilon \mid X = U \mid \varnothing = \varepsilon$ *(сохранение ограничением пустого отношения и пустого множества);*
4) $U \mid X = U \mid (X \bigcap \pi_1^2 U)$, $U = U \mid \pi_1^2 U$*; в частности,* $\pi_1^2 U \subseteq X \Rightarrow U \mid X = U$*;*
5) $(U \mid X) \mid Y = U \mid (X \bigcap Y)$*, в операторном виде* $\uparrow Y \circ \uparrow X = \uparrow (X \bigcap Y)$ *(композиция ограничений);*
   в частности, $\uparrow X \circ \uparrow X = \uparrow X$ *(идемпотентность оператора* $\uparrow X$*);*
6) $U \mid X \subseteq U$ *(оператор* $\uparrow X$ *убывающий);*
7) *оператор* $\uparrow X$ *является оператором замыкания (т.е. монотонным, убывающим и идемпотентным оператором) относительно теоретико-множественного включения* $\subseteq$*;*
8) $(\bigcup\limits_i U_i) \mid X = \bigcup\limits_i (U_i \mid X)$, $U \mid \bigcup\limits_i X_i = \bigcup\limits_i (U \mid X_i)$ *(дистрибутивность ограничения относительно объединений);*
9) $(\bigcap\limits_i U_i) \mid X = \bigcap\limits_i (U_i \mid X)$, $U \mid \bigcap\limits_i X_i = \bigcap\limits_i (U \mid X_i)$ *(дистрибутивность ограничения относительно пересечений);*
10) $\alpha \subseteq \beta \wedge X \subseteq dom\alpha \Rightarrow \alpha \mid X = \beta \mid X$*; в частности,* $\alpha \subseteq \beta \Rightarrow \alpha = \beta \mid dom\alpha$, $\alpha \subseteq \beta \wedge dom\alpha = dom\beta \Rightarrow \alpha = \beta$*;*

Очевидно, что операция наложения, вообще говоря, некоммутативная. Таким образом, возникает вопрос о поиске соответствующего критерия. Ответ приведен в следующем утверждении.

Предложение 2 (критерий коммутативности операции наложения). *Для произвольных функциональных бинарных отношений* α *и* β *выполняется эквивалентность* $\alpha \approx \beta \Leftrightarrow \alpha \nabla \beta = \beta \nabla \alpha$. $\square$

Следствие 1. *Пусть* α, β *–– функциональные бинарные отношения. Тогда имеют место следующие эквивалентности:*
$\alpha \nabla \beta = \beta \nabla \alpha \Leftrightarrow \alpha \nabla \beta = \alpha \bigcup \beta$
$\alpha \nabla \beta = \beta \nabla \alpha \Leftrightarrow \beta \nabla \alpha = \beta \bigcup \alpha$. $\square$

Следствие 2 (критерий совместности функций). *Пусть* α, β *—- функциональные бинарные отношения. Тогда следующие утверждения эквивалентны:*

1) $\alpha \cup \beta$ *— функциональное бинарное отношение;*
2) $\alpha \nabla \beta = \beta \nabla \alpha$*;*
3) $\alpha \nabla \beta = \alpha \cup \beta$*;*
4) $\alpha \approx \beta$*;*
5) $\beta \nabla \alpha = \beta \cup \alpha$. $\square$

Доказательство. Доказательство проводится с использованием свойства отношения совместности $\approx$ [31]: $\alpha \approx \beta \Leftrightarrow \alpha \bigcup \beta$ функциональное бинарное отношение.

Для проведения доказательств последующих утверждений важна следующая лемма.

Лемма 1. *Пусть* α, β *—- функциональные бинарные отношения. Тогда* $\alpha \bigcap \beta = (\alpha \bigcap \beta) \mid (dom\alpha \bigcap dom\beta)$. $\square$

Следствие 3. *Для функциональных бинарных отношений* α, β *выполняются две эквивалентности*
$\alpha \approx \beta \Leftrightarrow dom(\alpha \bigcap \beta) = dom\alpha \bigcap dom\beta$,
$\alpha \not\approx \beta \Leftrightarrow dom(\alpha \bigcap \beta) \subset dom\alpha \bigcap dom\beta$. $\square$

Следствие 4. *Для функциональных бинарных отношений* α, β *выполняются две эквивалентности*
$\alpha \approx \beta \Leftrightarrow \alpha \bigcap \beta = \alpha \mid (dom\alpha \bigcap dom\beta)$,
$\alpha \approx \beta \Leftrightarrow \alpha \bigcap \beta = \beta \mid (dom\alpha \bigcap dom\beta)$. $\square$

Следующее следствие дополняет следствие 2.

Следствие 5 (критерий совместности функций). *Пусть* α, β *—- функциональные бинарные отношения. Тогда следующие утверждения эквивалентны:*

1) $\alpha \approx \beta$*;*
2) $dom(\alpha \bigcap \beta) = dom\alpha \bigcap dom\beta$*;*
3) $\alpha \bigcap \beta = \alpha \mid (dom\alpha \bigcap dom\beta)$*;*
4) $\alpha \bigcap \beta = \beta \mid (dom\alpha \bigcap dom\beta)$. $\square$*;*

Что касается самой операции наложения, то ее свойства приведены в следующем утверждении.

Предложение 3 (свойства наложения). *Пусть* α, β *—- функциональные бинарные отношения. Тогда имеют место следующие утверждения:*

1) $\alpha \nabla \alpha = \alpha$ *(идемпотентность);*
2) $\alpha \nabla \beta \neq \beta \nabla \alpha$*, вообще говоря;*
3) $\alpha \nabla \beta = \beta \nabla \alpha \Leftrightarrow \alpha \approx \beta$ *(критерий коммутативности);*
4) $(\alpha \nabla \beta) \nabla \gamma = \alpha \nabla (\beta \nabla \gamma)$ *(ассоциативность).* $\square$

Доказательство. Докажем только 4 пункт. Будем использовать очевидное равенство $dom(\alpha\nabla\beta) = dom\alpha \bigcup dom\beta$. Используя определение наложения и свойства ограничения (предложение 1) имеем цепочку равенств:

$$
\begin{aligned}
&(\alpha\nabla\beta)\nabla\gamma = (\beta \cup \alpha \mid (dom\alpha \setminus dom\beta))\nabla\gamma = \\
&= \gamma \cup (\beta \cup \alpha \mid (dom\alpha \setminus dom\beta)) \mid (dom\alpha \cup dom\beta) \setminus dom\gamma = \\
&= \gamma \cup (\beta \cup \alpha \mid (dom\alpha \setminus dom\beta)) \mid ((dom\alpha \setminus dom\gamma) \cup (dom\beta \setminus dom\gamma)) = \\
&= \gamma \cup \beta \mid (dom\alpha \setminus dom\gamma) \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid (dom\alpha \setminus dom\beta) \mid \\
&\mid (dom\alpha \setminus dom\gamma) \cup (\alpha \mid (dom\alpha \setminus dom\beta)) \mid (dom\beta \setminus dom\gamma) = \\
&= \gamma \cup \beta \mid (dom\alpha \setminus dom\gamma) \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid \\
&\mid (dom\alpha \setminus dom\beta) \cap (dom\alpha \setminus dom\gamma) \cup \alpha \mid (dom\alpha \setminus dom\beta) \cap (dom\beta \setminus dom\gamma) = \\
&= \gamma \cup \beta \mid (dom\alpha \setminus dom\gamma) \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid \\
&\mid dom\alpha \setminus (dom\beta \cup dom\gamma) \cup \varepsilon = \gamma \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid \\
&\mid (dom\alpha \setminus (dom\beta \cup dom\gamma)) \cup \beta \mid (dom\alpha \setminus dom\gamma) = \\
&\qquad = \gamma \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid (dom\alpha \setminus (dom\beta \cup dom\gamma)). \quad (1)
\end{aligned}
$$

Прокомментируем неочевидные переходы. Переход от первого выражения ко второму и от второго к третьему сделали по определению наложения; при переходе от третьего к четвертому выражению воспользовались стандартным теоретико-множественным свойством $(A \cup B) \setminus C = A \setminus C \cup B \setminus C$; при переходе от четвертого к пятому выражению — дистрибутивностью ограничения относительно объединений (предложение 1); при переходе от пятого к шестому — свойством ограничения $(\alpha \mid A) \mid B = \alpha \mid (A \cap B)$ (предложение 1); при переходе от шестого к седьмому — теоретико-множественным равенством $(A \setminus B) \cap (A \setminus C) = A \setminus (B \cup C)$ и свойством ограничения $\alpha \mid ((dom\alpha \setminus dom\beta) \cap (dom\beta \setminus dom\gamma)) = \alpha \mid \varnothing = \varepsilon$ (предложение 1); при переходе от восьмого к девятому — свойствами ограничения (предложение 1)
$\alpha \mid B = \alpha \mid (dom\alpha \cap B)$,
$A \subseteq B \Rightarrow \alpha \mid A \subseteq \alpha \mid B$ и
$\beta \mid dom\alpha \setminus dom\gamma = \beta \mid dom\beta \cap (dom\alpha \setminus dom\gamma) \subseteq \beta \mid dom\beta \setminus dom\gamma$.
Перейдем к правой части равенства:

$$
\begin{aligned}
&\alpha\nabla(\beta\nabla\gamma) = \alpha\nabla(\gamma \cup \beta \mid (dom\beta \setminus dom\gamma)) = \\
&\qquad = \gamma \cup \beta \mid (dom\beta \setminus dom\gamma) \cup \alpha \mid (dom\alpha \setminus (dom\beta \cup dom\gamma)). \quad (2)
\end{aligned}
$$

Из (1), (2) и вытекает доказываемое равенство. □

Пусть F — класс всех функциональных бинарных отношений (на некотором зафиксированном универсуме D). Очевидно, что $< F, \subseteq >$ — частично упорядоченное множество (ч. у. м.). Перейдем к установлению его структуры. Здесь имеют место два следующих утверждения (для ч. у. м. используем терминологию [32]).

Предложение 4. *Ч. у. м.* $< F, \subseteq >$ *есть нижняя полурешетка, причем* $inf\{f, g\} = f \cap g$. □

Доказательство. Доказательство следует из того, что $< F, \cap >$ есть коммутативная идемпотентная полугруппа, и хорошо известной связи между такими полугруппами и нижними полурешетками,(см., например, [32]).

Таким образом, в терминах классов наибольшая общая часть двух классов существует всегда, но она может быть пустой. Более полную информацию о ч. у. м. $< F, \subseteq >$ даёт следующее утверждение.

Предложение 5 (структура ч. у. м. $< F, \subseteq >$). *Выполняются следующие утверждения:*

1) *Всюду неопределённая функция $f_\varnothing$ — наименьший элемент ("дно") ч. у. м. $< F, \subseteq >$.*
2) *Наибольший элемент в ч. у. м. $< F, \subseteq >$ существует тогда и только тогда, когда универсум D — не больше чем одноэлементный.*
3) *Точная нижняя грань (инфимум) существует для любого непустого множества $\mathcal{F}$, причем $inf\,\mathcal{F} = \cap_{f \in \mathcal{F}} f$.*
4) *Точная верхняя грань множества $\mathcal{F}$ (супремум) существует тогда и только тогда, когда множество $\mathcal{F}$ ограничено сверху, при этом $sup\,\mathcal{F} = \cup_{f \in \mathcal{F}} f$.*
5) *Элемент f является атомом тогда и только тогда, когда график f — одноэлементный, т.е. f есть функцией вида $f : \{x\} \to \{y\}$.*
6) *Ч. у. м. $< F, \subseteq >$ является условно полным ч. у. м. и полной (верхней) полурешеткой (complete semilattice).* $\square$

Так как отношение наследования $\leq$ на классах вводится покомпонентно (эквивалентно: операцией прямого произведения порядков), то свойства ч. у. м. $< F, \subseteq >$ переносятся на ч. у. м. $< \mathbb{K}, \leq >$: например, $< f_\varnothing, f_\varnothing >$ — наименьший элемент и т.д.

Приведем соответствующий классический результат (см., например, [33]). Пусть $< D_1, \leq_1 >$, $< D_2, \leq_2 >$ — два ч. у. м. Их прямое произведение определяется стандартно: $< D_1 \times D_2, \preceq >$, где $< d_1, d_2 > \preceq < d_1', d_2' > \Leftrightarrow d_1 \leq_1 d_1' \wedge d_2 \leq_2 d_2'$. Для точных граней имеют место формулы:

$$\underset{D_1 \times D_2}{sup}\, L \simeq < \underset{D_1}{sup}\, \pi_1^2 L, \underset{D_2}{sup}\, \pi_2^2 L >, \tag{3}$$

$$\underset{D_1 \times D_2}{inf}\, L \simeq < \underset{D_1}{inf}\, \pi_1^2 L, \underset{D_2}{inf}\, \pi_2^2 L >, \tag{4}$$

$L \subseteq D_1 \times D_2$.

Здесь $\simeq$ обобщенное равенство (сильное равенство Клини). Формулы (3) и (4) и позволяют переносить свойства ч. у. м. $< F, \subseteq >$ на ч. у. м. $< \mathbb{K}, \leq >$.

Приведённые формальные результаты могут иметь практическое значение при анализе диаграмм классов. Это во-первых, выделение компонент связности в соответствующем графе (по терминологии [32] речь идет о разложении ч. у. м. в кардинальную сумму). Выделение компонент соответствует декомпозиции системы на подсистемы. Во-вторых, в диаграмме классов можно

проводить поиск клонов для их элиминации (по этой проблематике см. [29]). Наконец, в-третьих, диаграмму классов можно подвергнуть модификации с помощью операций сочленения и пересечения с целью оптимизации по соответствующему критерию.

5. Результаты, выводы

В работе приведен короткий анализ доступной библиографии по ООП, уточняются классы ООП в виде пар функциональных бинарных отношений. Бинарное отношение, являющееся первой компонентой, атрибутам приписывает их типы (значения); отношение, являющееся второй компонентой, сигнатурам методов приписывает их семантику (тип значений функции — семантики для интерфейса).

Над классами рассматриваются операции пересечения и сочленения. Пересечение классов вводится на основе стандартного теоретико-множественного пересечения бинарных отношений, а сочленение — на основе специальной операции наложения, которая позволяет разрешить конфликт имен. Указанные операции над классами могут быть использованы при рефакторинге (refactoring) программ.

Наследование классов уточняется в виде стандартного теоретико-множественного включения между соответствующими компонентами классов. В работе получены следующие основные математические результаты:

- установлены основные свойства наложения (идемпотентность, ассоциативность, критерий коммутативности в терминах отношения совместности);
- структура ч. у. м. множества всех функциональных бинарных отношений, упорядоченного по включению (это ч. у. м. является нижней полурешеткой; всюду неопределённая функция является наименьшим элементом; инфимумы непустых подмножеств существуют всегда, супремумы подмножеств существуют тогда и только тогда, когда подмножества ограничены с верху; для точных граней приведены формулы их нахождения).

Предложенная формальная модель может использоваться для анализа и модификации диаграмм классов:

- нахождения компонент связности, что соответствует декомпозиции системы на подсистемы;
- поиска клонов;
- модификации диаграмм с помощью введенных операций пересечений и сочленения.

Что касается дальнейших математических исследований, имеющих содержательную интерпретацию, то это, например, установление взаимной дистрибутивности рассматриваемых операций.

Список литературы

1. Parnas, D. On the Criteria to Be Used in Decomposing Systems into Modules, in Classics in Software Engineering. Magazine Communications of the ACM, Volume 15, Issue 12, Dec. 1972, p. 1053–1058 (1972)
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд. : Пер. с англ. Москва: ООО “И.Д. Вильямс”, 2008. —- 720 с. (2008)
3. Лаврищева Е. М. Методы программирования: теория, инженерия, практика. Киев: Наукова думка, 2006. —- 451 с. (2006)
4. Wirfs-Brock R. Designing Object-Oriented Software. Prentice Hall, New Jersey, 1990. —- 341 p. (1990)
5. Шлеер С. Объектно-ориентированный анализ: моделирование мира в состояниях. Киев: Диалектика, 1993. —- 238 с. (1993)
6. Бадд Т. Объектно-ориентированное программирование в действии. Питер, 1997. — 464 с. (1997)
7. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001. —- 368 с. (2001)
8. Мухортов В. В. Объектно-ориентированное программирование, анализ и дизайн. Методическое пособие. Новосибирск, 2002. — 108 с. (2002)
9. Грэхем И. Объектно-ориентированные методы. Принципы и практика. 3-е издание. Москва: ООО “И.Д. Вильямс”, 2004. —- 880 с. (2004)
10. Лаврищева Е. М. Сборочное программирование. Основы индустрии программных продуктов. Киев: Наукова думка, 2009. — 371 с. (2009)
11. Фридман А. Л. Основы объектно-ориентированной разработки программных систем. Москва: “Финансы и статистика”, 2000. — 192 с. (2000)
12. Лаврищева Е. М. Интерфейс в программировании. Киев: Проблеми програмування, №2, 2007. С. 126–139 (2007)
13. The Common Object Request Broker: Architecture and Specification -– OMG, www.omg.org/spec/CORBA/3.3/
14. Безопасность критических инфраструктур: математические и инженерные методы анализа и обеспечения / под ред. Харченко В. С. – Министерство образования и науки Украины, Национальный аэрокосмический университет им. Н. Е. Жуковского “ХАИ”, 2011. — 641 с. (2011)
15. Пискунов А. Г. Формализация парадигмы объектно-ориентированного программирования, www.realcoding.net/dn/docs/machine.pdf
16. Richta, Karel,Toth, David. Formal Models of Object-Oriented Databases. In Objekty 2008. Žilina: Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, 2008, p. 204-217, www.ksi.mff.cuni.cz/ richta/publications/richta-toth-Objekty2008.pdf
17. Buy, Dmitriy, Kompan, Sergiy. The Concepts of Object, Class, Inheritance, Life Cycle: Formalization. First International Workshop “Critical infrastructure safety and security” (CrISS-Dessert’11), Kirovograd, Ukraine, May 11-13, 2011, p. 236-244 (2011)
18. Буй Д. Б., Компан С. В. Уточнення множинного успадкування у вигляді операції накладання. Вісник Київського національного університету імені Тараса Шевченка, Серія: Фізико-математичні науки, випуск №4, 2012, с. 111-119 (2012) (на украинском языке)
19. Редько В. Н. Композиции программ и композиционное программирование. Программирование. — 1978. — № 5. С. 3-24 (1978)

20. Редько В. Н. Основания композиционного программирования. Программирование. — 1979. — № 3. С. 3-13. (1979)
21. Редько В. Н. Экзистенциальные основания композиционной парадигмы. Кибернетика и системный анализ. — 2008. — № 2. С. 3-12 (2008)
22. Object Data Management Group. http://www.odbms.org/odmg/
23. Sarkar, Manojit, Reiss, Steven P. A Data Model and A Query Language for Object-Oriented Database. Island, Department of Computer Science Brown University Providence, Rhode, December 1992, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.4531&rep=rep1&type=pdf (1992)
24. Shaw Gail M., Zdonik Stanley B. A Query Algebra for Object-Oriented Databases. Island, Department of Computer Science. Brown University Providence, Rhode, March 1989, trac.common-lisp.net/elephant/raw-attachment/wiki/RelationalAlgebra/shaw89query.2.pdf
25. Suri, Pushpa R., Rani, Sudesh. Database Algebras. Journal of Theoretical and Applied Information Technology, 2005, p. 595-602, www.jatit.org/volumes/research-papers/Vol4No7/7.pdf
26. Буй Д. Б., Компан С. В. Операции объединения и пересечения спецификаций классов в многосортной алгебраической системе для объектно-ориентированного программирования. Сборник научных трудов SWorld. Материалы международной научно-практической конференции “Современные проблемы и пути их решения в науке, транспорте, производстве и образовании’2012”. — Выпуск 4. Том 3. — Одесса: КУПРИЕНКО, 2012. —- ЦИТ: 412-1264, с. 45-49 (2012)
27. Roy C. K. A survey on software clone detection research // SCHOOL OF COMPUTING TR 2007-541, QUEEN’S UNIVERSITY. 2007. — Vol. –115. (2007)
28. Фаулер М. Рефакторинг. Улучшение существующего кода. — Символ-Плюс, 2008. (2008)
29. Булычев П. Е. Алгоритмы вычислений подобия в задачах верификации и реструктуризации программ. Диссертация на соискание ученой степени кандидата физико-математических наук: спец. 05.13.11 “Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей” — Москва: 2010. — 169 с. (2010)
30. Буй Д. Б., Кахута Н. Д. Властивості теоретико-множинних конструкцій повного образу та обмеження. — Вісник Київського університету ім. Т. Шевченка. Сер. “Фіз.-мат. науки”, 2005. — Вип. 2. С. 157-170 (2005) (на украинском языке)
31. Буй Д. Б. Властивості відношення конфінальності та устрій множини часткових функцій. / Д. Б. Буй, Н. Д. Кахута // Вісник Київського університету ім. Т. Шевченка. Сер. “Фіз.-мат. науки”, 2006. — Вип. 2. С. 125-135 (2006) (на украинском языке)
32. Скорняков Л. А. Элементы теории структур. — Москва: “Наука”, 1982. — 158 с. (1982)
33. Burbaki. N. Elements de Mathematique. Premiere Partie. Les Structures Fondamentals de L’Analise. Livre 1. Theorie Des Ensembles. Chapter III, section 1. 1958 (1958)