УДК 004.655

©2013. **Д. Б. Буй, С. В. Компан**

ДИАГРАММЫ КЛАССОВ ООП: ФОРМАЛИЗАЦИЯ И АНАЛИЗ

Дан сравнительный анализ работ, посвящённых формальным моделям объектно-ориентированного программирования (ООП). Предмет исследования – модель диаграммы классов (соответствующее частично упорядоченное множество). Модель класса (спецификации класса) – пара функциональных бинарных 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, что позволяет выделять подсистемы, клоны и оптимизировать по соответствующему критерию саму диаграмму.

Ключевые слова: *объектно-ориентированное программирование (ООП), формальные модели ООП, диаграмма классов, операция сочленения классов, операция пересечения классов, клон.*

1. Введение. Основы объектно-ориентированного программирования (ООП) были заложены в конце 60-х годов прошлого столетия. В основе ООП лежит объектная модель данных (ОМД). Появление ОМД связано с появлением абстрактных типов данных. В 1971 году D.L. Parnas в [1] предложил идею инкапсуляции (сокрытия информации). В 80-х годах G. Booch в [2, 3] 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 ПС, в основе которого лежат принципы ООП, и 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 [4]. На сегодняшний день не существует единого мнения о том, что лежит в основе объектной модели (ОМ). С одной стороны ОМ, определяют такие принципы: абстракция, инкапсуляция, наследование и полиморфизм [5, 6], и этот подход поддерживает большинство профессионалов. Язык программирования, в котором реализованы эти принципы, является объектно-ориентированным, и именно они придают языку гибкость при программной реализации сущностей ПрО. Наряду с этим, существует и другое мнение. В частности, в основе ОМ лежит только механизм наследования. Этой точ-

ки зрения придерживается L. Cardelli [7]. Кроме того, например, на форуме сайта www.sql.ru можно найти взгляд одного из пользователей: «Определение того, что делает язык объектно-ориентированным, остаётся спорным. Исследование разницы между Simula, Smalltalk и другими языками подсказывает мысль, что наследование является единственной критической идеей, ассоциирующейся с ООП. Сопрограммы, передача сообщений, статическая/динамическая область видимости, проверка типов, простые/множественные родительские классы – все эти явно независимые характеристики могут присутствовать или отсутствовать в языке, который считается обьектно-ориентированным. Поэтому теория ООП должна прежде всего сосредоточиться на значении термина наследование» [8]. Авторы этой статьи согласны, что наследование классов придаёт мощь и гибкость объектно-ориентированному языку программирования, причем реализация наследования, как правило, предполагает использования полиморфизма и инкапсуляции, то есть принципы полиморфизма и инкапсуляции неявно присутствуют в термине «наследование». Но вместе с тем придерживаются другого мнения по поводу того, что именно делает язык объектно-ориентированным; аргументация такая: например, в функциональном языке Haskell реализовано наследование, но это вовсе не говорит о том, что этот язык является объектно-ориентированным. Соответственно, утверждение о том, что только наследование определяет ОМ, является неполным и поэтому некорректным.

2. Краткий анализ современного состояния ООП. В связи с популярностью ООП как средства проектирования и разработки ПС существует достаточное количество соответствующей литературы [2, 3, 4, 5, 9, 10, 11, 12, 13, 14, 15]. Следует отметить ряд работ Е.М. Лаврищевой [4, 14, 16], в которых уделяется большое внимание проектированию и программированию ПС, в основе которых лежит ООП. Автор предлагает выделять четыре уровня абстрактного представления ОМ: обобщающий, структурно-упорядоченный, характеристический и поведенческий. На каждом из этих уровней применяется свой математический аппарат [14].

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 [17]. В данной модели для описания понятий применяется декларативный язык определения интерфейсов IDL (Interface Definition Language).

Для полной гарантии того, что информационная система, построенная на основе ООП, будет удовлетворять исходным спецификациям и стабильно работать (будет гарантоустойчивой по терминологии [18]), 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 [19, 20] формально даётся определение основных понятий ООП: класса, объекта, наследования, инкапсуляции, полиморфизма с использованием математического аппарата теории множеств,

функций, абстрактных автоматов Мили и Мура (в этой же работе приводится обширная библиография по ООП). В [21] авторы строят формальную модель для объектных баз данных (БД), в основе которой лежит теория категорий. В [22, 23] дано формальное определение основных понятий ООП, в основе определения которых лежит композиционное программирование, созданное академиком НАН Украины В.Н. Редько [24, 25, 26].

Особое внимание следует уделить вопросу построения ОМ 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. ОМ 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 [27].

В [28] 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). Авторы приводят синтаксическое описание класса (неформальное). Даётся формальное определение метода, который является функцией, отображающей сигнатуру в тело функции. Сигнатура описывается следующим образом: $c : m(\tau_1, \tau_2, \ldots, \tau_n) \rightarrow \tau_r$, 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, υ – 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 [28, 29, 30]. Различаются эти алгебры только сигнатурными операциями над объектами. В [31] авторы вводят операции пересечения и сочленения (в работе она называлась объединением) над классами. В результате предлагается рассматривать вместо объектной алгебры алгебраическую систему $< O, K; \Omega_{obj}; \Omega_{spec}, \leq >$, 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. В [32] обсуждаются выделение типа (Subtyping) и наследование (Inheritance) с использованием алгебр и коалгебр. В [33] обсуждаются такие понятия, как: тип, класс, подкласс, выделение типа (Subtyping), наследование (Inheritance) и их отличия.

3. Постановка задачи. Построение новой сложной ПС, как правило, сопряжено с нетривиальными усилиями. В результате для решения этой задачи 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 коллектива разработчиков во времени приводит к тому, что для проверки правильности программ выбирается тестирование вместо построения формальной модели и доказательства правильности ее работы. 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, при которой в новую систему могут попасть «одинаковые» (в том или ином смысле) части кода (атрибуты и методы) – т.е. 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.

Приведем репрезентативный пример. В примере будет использована нотация языка программирования Java.

Пусть необходимо написать программу «Учебное заведение», которая может использоваться как в школе, так и в ВУЗе. 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 результате получим новую диаграмму классов (рис. 3), относительно которой сделаем несколько замечаний.

Как указывается в [34], клоны могут быть удалены с использованием операций «выделение метода» (Extract Method) и «Подъем метода» (Pull Up Method), изложенных в [35].

Выделение метода (т.е. процедурная абстракция) заключается в выделении общего кода в новую функцию (или метод класса, если речь об ООП).

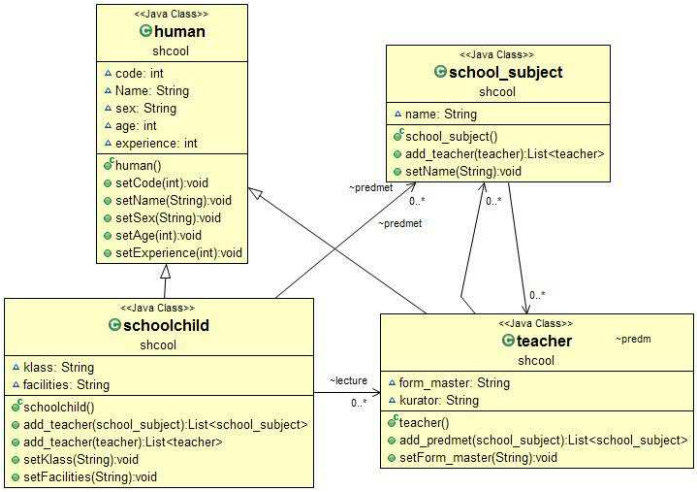

Рис. 1. Диаграмма классов ПрО "Школа"

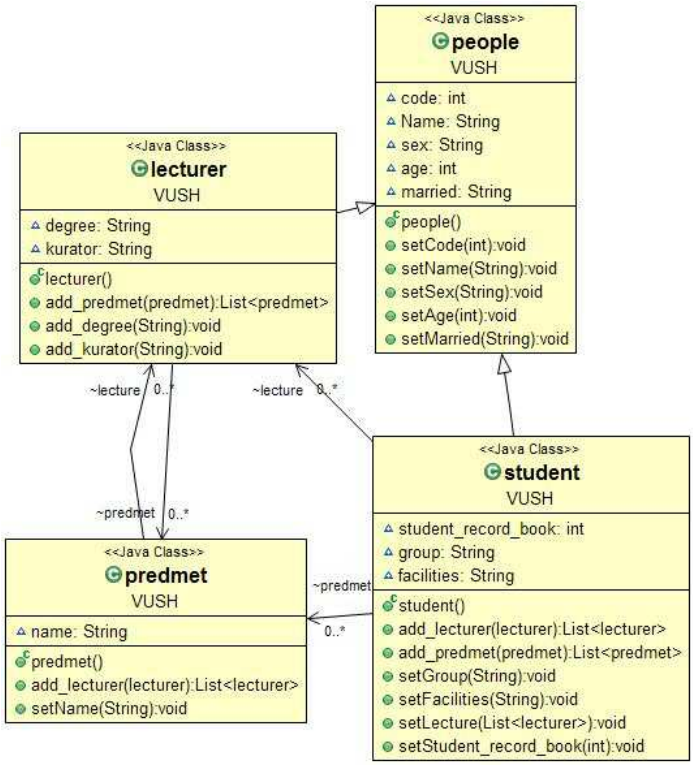

Рис. 2. Диаграмма классов ПрО "ВУЗ"

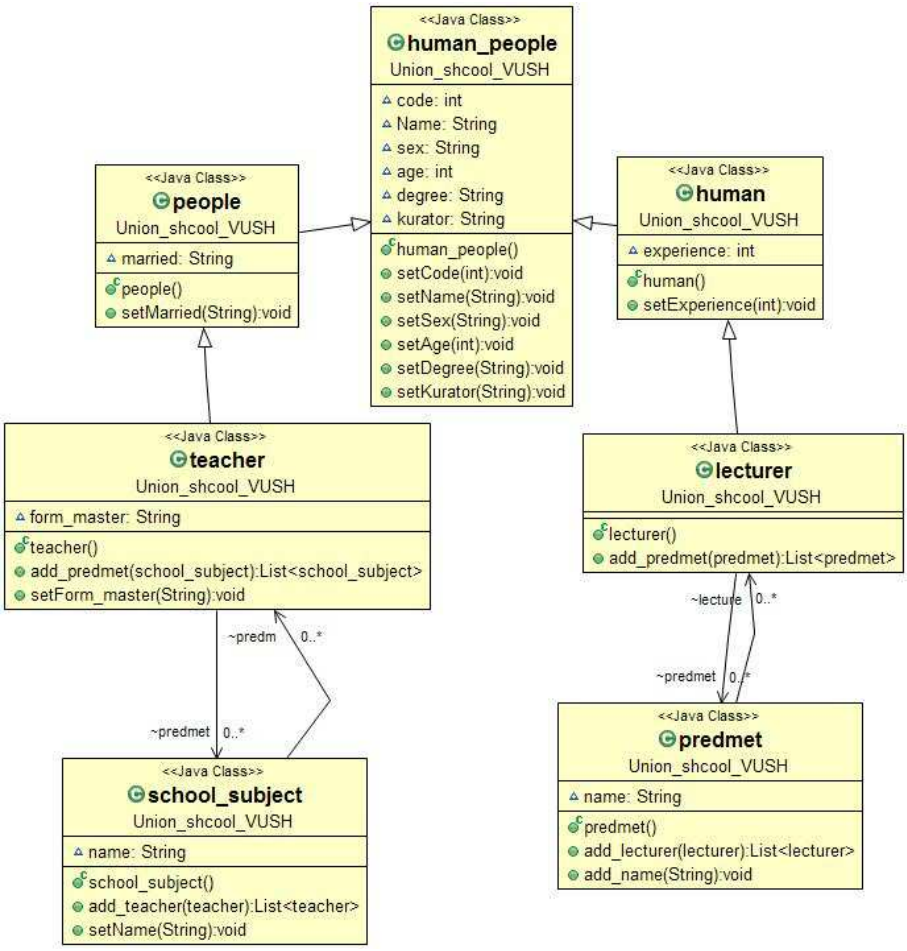

Рис. 3. Диаграмма классов ПрО "Школа и ВУЗ"

Подъем метода заключается в перемещении общего метода 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) [36]. В нашем подходе могут быть также выделены и общие атрибуты. Фактически своими преобразованиями над классами мы изменяем внутреннюю структуру программ, не затрагивая их поведения и в какой-то степени улучшаем понимание кода.

Целью работы является разработка фрагмента 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 атрибутам ставит в соответствие их типы (множества значений из универсального домена D), а μ – функциональное бинарное отношение, которое сигнатурам методов ставит в соответствие их реализацию (семантику, логику). Можно рассматривать μ как функциональное бинарное отношение, которое сигнатуре метода ставит в соответствие его тип; в этом случае получим определение интерфейса. 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 далее.

Определение. 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 на множество X, т.е. $\gamma \mid X = \gamma \bigcap (X \times range\ \gamma)$ ($range\ \gamma$ – проекция отношения γ по второй компоненте).

Наложение иллюстрируется на рис. 4.

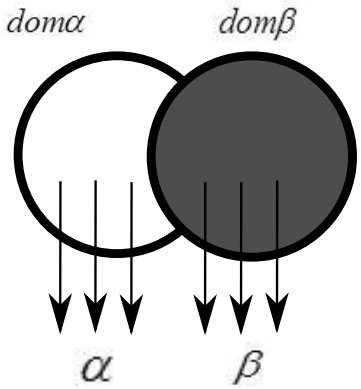

Рис. 4. Графическая интерпретация операции наложения $\alpha \nabla \beta$ (β имеет приоритет)

Введем бинарные операции 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 $< s_1, \mu_1 > \coprod < s_2, \mu_2 > = < s_1 \nabla s_2, \mu_1 \nabla \mu_2 >$.

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. $doms_1 \bigcap doms_2 \neq \varnothing$ и/или $dom\mu_1 \bigcap dom\mu_2 \neq \varnothing$. Тогда возникает конфликт имен и нужно определить по определённому 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 (родительского, 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. $range\ \mu$) является множеством множеств, которые интерпретируются как области значений функций-семантик методов, то получим определение интерфейса.

Определение. 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 принимают одинаковые значения на общих аргументах.

При доказательстве свойств операций над классами и установлении структуры семейства классов будут использоваться следующие абстрактные свойства ограничения [37]. Ниже $U_1, U_2, \ldots$ – произвольные бинарные отношения, $X, X_1, \ldots$ – произвольные множества, $\alpha, \beta, \ldots$ – как и ранее, произвольные функциональные бинарные 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, при этом $\pi_1^2\alpha = dom\ \alpha$ и $\pi_2^2\alpha = range\ \alpha$.

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 по первой компоненте 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. $(U \mid X) \mid Y = U \mid (X \bigcap Y)$ или 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, $\alpha \subseteq \beta \Rightarrow \alpha = \beta \mid dom\alpha$, $\alpha \subseteq \beta \wedge dom\alpha = dom\beta \Rightarrow \alpha = \beta$.

Операция наложения некоммутативная. Критерий коммутативности наложения имеет следующий вид.

Предложение 2. Для произвольных функциональных бинарных отношений α и β выполняется эквивалентность $\alpha \approx \beta \Leftrightarrow \alpha \triangledown \beta = \beta \triangledown \alpha$.

Следствие 1. *Если α, β – произвольные функциональные бинарные отношения, то истинны следующие эквивалентности: $\alpha \triangledown \beta = \beta \triangledown \alpha \Leftrightarrow \alpha \triangledown \beta = \alpha \bigcup \beta$ и $\alpha \triangledown \beta = \beta \triangledown \alpha \Leftrightarrow \beta \triangledown \alpha = \beta \bigcup \alpha$.*

Следствие 2 . *Пусть α, β – произвольные функциональные бинарные отношения. Тогда следующие утверждения эквивалентны:*

1. *$\alpha \cup \beta$ — функциональное бинарное отношение;*

2. *$\alpha \triangledown \beta = \beta \triangledown \alpha$;*

3. *$\alpha \triangledown \beta = \alpha \cup \beta$;*

4. *$\alpha \approx \beta$;*

5. *$\beta \triangledown \alpha = \beta \cup \alpha$.*

Доказательство проводится с использованием свойства отношения совместности $\approx$: $\alpha \approx \beta \Leftrightarrow \alpha \bigcup \beta$ функциональное бинарное отношение [38].

Для доказательств последующих утверждений важна следующая лемма.

Лемма. *Пусть α, β – произвольные функциональные бинарные отношения. Тогда $\alpha \bigcap \beta = (\alpha \bigcap \beta) \mid (dom\alpha \bigcap dom\beta)$.*

Следствие 3. *Для произвольных функциональных бинарных отношений α, β истинны следующие эквивалентности: $\alpha \approx \beta \Leftrightarrow dom(\alpha \bigcap \beta) = dom\alpha \bigcap dom\beta$ и $\alpha \not\approx \beta \Leftrightarrow dom(\alpha \bigcap \beta) \subset dom\alpha \bigcap dom\beta$.*

Следствие 4. *Для функциональных бинарных отношений α и β истинны следующие две эквивалентности:*

$$\alpha \approx \beta \Leftrightarrow \alpha \bigcap \beta = \alpha \mid (dom\alpha \bigcap dom\beta),$$

$$\alpha \approx \beta \Leftrightarrow \alpha \bigcap \beta = \beta \mid (dom\alpha \bigcap dom\beta).$$

Следующее следствие дополняет следствие 2.

Следствие 5 (критерии совместности функций). *Если α, β – произвольные функциональные бинарные отношения, то следующие утверждения эквивалентны:*

1. $\alpha \approx \beta$;
2. $dom(\alpha \bigcap \beta) = dom\alpha \bigcap dom\beta$;

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

Предложение 3. Пусть α, β – произвольные функциональные бинарные отношения. Тогда имеют место следующие утверждения:

1. $\alpha \triangledown \alpha = \alpha$ (идемпотентность);
2. $\alpha \triangledown \beta \neq \beta \triangledown \alpha$;
3. $\alpha \triangledown \beta = \beta \triangledown \alpha \Leftrightarrow \alpha \approx \beta$ (критерий коммутативности);
4. $(\alpha \triangledown \beta) \triangledown \gamma = \alpha \triangledown (\beta \triangledown \gamma)$ (ассоциативность).

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

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

Прокомментируем неочевидные переходы. Переход от первого выражения ко второму и от второго к третьему сделан по определению наложения; при переходе от третьего к четвертому выражению использовано стандартное теоретико-множественное свойство $(A \cup B) \setminus C = (A \setminus C) \cup (B \setminus C)$; при переходе от четвер-

того к пятому выражению – дистрибутивность ограничения относительно объединений (предложение 1); при переходе от пятого к шестому – свойство ограничения $(\alpha \mid X) \mid Y = \alpha \mid (X \cap Y)$ (предложение 1); при переходе от шестого к седьмому – теоретико-множественное равенство $(A \setminus B) \cap (A \setminus C) = A \setminus (B \cup C)$ и свойство 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 – свойство ограничения (предложение 1) $\alpha \mid X = \alpha \mid (dom\alpha \cap X)$, $X \subseteq Y \Rightarrow \alpha \mid X \subseteq \alpha \mid Y$ 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:

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

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

Исследуем структуру отношения наследования множества классов.

Начнем с структуры частично упорядоченного по включению множества функций. Пусть F – класс всех функциональных бинарных отношений (на некотором зафиксированном универсуме D). Очевидно, что $< F, \subseteq >$ – частично упорядоченное множество (ч. у. м.). Исследуем его структуру. Имеют место два следующих утверждения (для ч. у. м. использована терминология [39]).

Предложение 4. Ч. у. м. $< F, \subseteq >$ – нижняя полурешетка, а $inf\{f, g\} = f \cap g$.

Доказательство. Доказательство следует из того, что $< F, \cap >$ есть коммутативная идемпотентная полугруппа, и связи между такими полугруппами и нижними полурешетками (см., напр., [39]). □

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, причем $sup\,\mathcal{F} = \cup_{f \in \mathcal{F}} f$.

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).

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).

Приведем соответствующий классический результат (см., например, [40]). 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:

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

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

$$L \subseteq D_1 \times D_2,$$

где $\simeq$ – 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 имеют практическое значение при анализе диаграмм классов. Это, во-первых, выделение компонент связности в соответствующем графе (по терминологии [39] 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. [36]). 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 краткий анализ доступной библиографии по ООП, уточняются классы ООП в виде пар функциональных бинарных отношений. Бинарное отношение, являющееся первой компонентой, характеризуется тем, что атрибутам приписывает их типы; отношение, являющееся второй компонентой, характеризуется тем, что 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 посредством стандартного теоретико-множественного включения между соответствующими компонентами классов. В работе получены следующие основные математические результаты:

1) установлены основные свойства наложения (идемпотентность, ассоциативность, различные критерии коммутативности);

2) установлена 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 сверху; для точных граней приведены формулы их нахождения; эти же свойства имеет частично упорядоченное множество классов.

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

1) для нахождения компонент связности, что соответствует декомпозиции системы на подсистемы, а это важно при рефакторинге;

2) для поиска клонов с целью их элиминации;

3) для модификации диаграмм с помощью введенных операций пересечения и сочленения с целью оптимизации по соответствующему критерию.

Дальнейшие исследования, имеющие содержательную интерпретацию, состоят, например, в анализе взаимной дистрибутивности рассматриваемых операций.

1. *Parnas D.* On the Criteria to Be Used in Decomposing Systems into Modules, in Classics in Software Engineering // Magazine Communications of the ACM. – 1972. – Vol. 15, issue 12. – P. 1053–1058.
2. *Booch G.* Object-Oriented Development, IEEE Transactions on Software Engineering. – 1986. – **12**, No. 2. – P. 211–221.
3. *Booch G.* Object-Oriented Analysis and Design with Applications. – Redwood City, CA: Benjamin/Cummings. – 1991. – 691 p.
4. *Лаврищева Е.М.* Методы программирования: теория, инженерия, практика. – К.: Наукова думка, 2006. – 451 с.
5. *Мухортов В.В.* Объектно-ориентированное программирование, анализ и дизайн. Методическое пособие. – Новосибирск, 2002. – 108 с.
6. *Finkel A.* Advanced programing language design. – NY: Addison-Wesley, 1996. – 363 p.
7. *Cardelli L.* A Semantics of Multiple Inheritance. Information and Computation 76, 1988. – P. 138–164.
8. Точка входа: http://www.sql.ru/forum/379368-33/vpechatleniya-novichka-ot-oop.
9. *Wirfs-Brock R.* Designing Object-Oriented Software. – New Jersey: Prentice Hall, 1990. – 341 p.
10. *Шлеер С.* Объектно-ориентированный анализ: моделирование мира в состояниях. – К.: Диалектика, 1993. – 238 с.
11. *Бадд Т.* Объектно-ориентированное программирование в действии. – СПб: Питер, 1997. – 464 с.
12. *Гамма Э.* Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2001. – 368 с.
13. *Грэхем И.* Объектно-ориентированные методы. Принципы и практика. – М.: Вильямс, 2004. – 880 с.
14. *Лаврищева Е.М.* Сборочное программирование. Основы индустрии программных продуктов. – К.: Наукова думка, 2009. – 371 с.
15. *Фридман А.Л.* Основы объектно-ориентированной разработки программных систем. – М.: "Финансы и статистика", 2000. – 192 с.
16. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування. – 2007. – № 2. – С. 126–139.
17. The Common Object Request Broker: Architecture and Specification – OMG, www.omg.org/spec/CORBA/3.3.
18. Безопасность критических инфраструктур: математические и инженерные методы анализа и обеспечения / под ред. Харченко В.С. – Харьков: ХАИ, 2011. – 641 с.

19. *Пискунов А.Г.* Формализация парадигмы объектно-ориентированного программирования, www.realcoding.net/dn/docs/machine.pdf.
20. *Пискунов А.Г.* Formalization of the OOP paradigm: Inheritance of Abstract Automata / А.Г. Пискунов, С.М. Петренко // Вестник Киевского ун-та. Серия: Кибернетика, 2011. – Вып. 11. – С. 40–44.
21. *Richta K., Toth D.* Formal Models of Object-Oriented Databases. – www.ksi.mff.cuni.cz/richta/publications/richta-toth-Objekty2008.pdf.
22. *Buy D., Kompan S.* The Concepts of Object, Class, Inheritance, Life Cycle: Formalization // Proc. of the First International Workshop "Critical infrastructure safety and security" (CrlSS-Dessert'11). – 2011. – Kirovograd, Ukraine, May 11-13. – P. 236–244.
23. *Буй Д.Б., Компан С.В.* Уточнення множинного успадкування у вигляді операції накладання // Вісник Київського національного ун-ту ім. Тараса Шевченка. Серія: Фізико-математичні науки. – 2012. – Вип. 4. – С. 111–119.
24. *Редько В.Н.* Композиции программ и композиционное программирование // Программирование. – 1978. – № 5. – С. 3-24.
25. *Редько В.Н.* Основания композиционного программирования // Программирование. – 1979. – № 3. – С. 3–13.
26. *Редько В.Н.* Экзистенциальные основания композиционной парадигмы // Кибернетика и системный анализ. – 2008. – № 2. – С. 3–12.
27. Object Data Management Group, http://www.odbms.org/odmg.
28. *Reiss S.M.* A Data Model and A Query Language for Object-Oriented Database. – citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.4531&rep=rep1&type=pdf.
29. *Shaw G.M., Zdonik S.B.* A Query Algebra for Object-Oriented Databases. – trac.common-lisp.net/elephant/raw-attachment/wiki/RelationalAlgebra/shaw89query.2.pdf.
30. *Pushpa S.R., Sudesh R.* Database Algebras // Journal of Theoretical and Applied Information Technology. – 2005. – P. 595–602.
31. *Буй Д.Б., Компан С.В.* Операции объединения и пересечения спецификаций классов в многосортной алгебраической системе для объектно-ориентированного программирования. – Материалы международной научно-практической конференции "Современные проблемы и пути их решения в науке, транспорте, производстве и образовании 2012". – Одесса: КУПРИЕНКО, 2012. – ЦИТ: 412–1264. – Вып. 4, т. 3. – С. 45–49.
32. *Pool E.* Subtyping and Inheritance for Inductive Types // Proc. of the TYPES'97 Workshop on Subtyping, inheritance and modular development of proofs. – UK: Durham, 1997.
33. *Abadi M., Cardelli L.* A theory of objects. – Springer, 1996. – http://books.google.com.ua/books?id=4xT3LgCPP5UC.
34. *Roy C.K.* A survey on software clone detection research // School of computing TR 2007–541. – UK: Queen's University. – 2007. – Vol. 115.
35. *Фаулер М.* Рефакторинг. Улучшение существующего кода. – Символ-Плюс, 2008.
36. *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". – М.: 2010. – 169 с.
37. *Буй Д.Б., Кахута Н.Д.* Властивості теоретико-множинних конструкцій повного образу та обмеження // Вісник Київського ун-ту ім. Тараса Шевченка. Сер. "Фіз.-мат. науки", 2005. – Вип. 2. – С. 232–240.
38. *Буй Д.Б., Кахута Н.Д.* Властивості відношення конфінальності та устрій множини часткових функцій // Вісник Київського ун-ту ім. Тараса Шевченка. Сер. "Фіз.-мат. науки", 2006. – Вип. 2. – С. 125–135.
39. *Скорняков Л.А.* Элементы теории структур. – М.: Наука, 1982. – 160 с.
40. *Бурбаки Н.* Теория множеств. – М.: Мир, 1965. – 455 с.

D. B. Buy, S. V. Kompan

OOP class diagrams: formalization and analysis.

The article deals with comparative analysis of the papers dedicated to formal models of object-oriented programming (OOP). The subject of investigation is a model of a class diagram (a corresponding partially ordered set). A class model (class specifications) is a pair of functional binary relations, one component specifies the attributes, the second one – the methods. Inheritance is defined as inclusion of function graphs. Classes are subjected to the operations of intersection and junction. The junction and the intersection of class speci?cations is important as it provides for the opportunity to construct the core of a new program with two programs which allows integrating these two programs that results in the Framework version. The formal results are as follow: the structure of partially ordered set of classes, properties overlapping operation, in terms of which the operation of junction is defined: idempotency,associativity, criterion of commutativity. Model class diagram can be used to analyze the structure of classes that allows to find subsystems, clones and optimize the chart itself according to the relevant criterion.

Keywords: *object-oriented programming, formal models of OOP, the class diagram, junction of class specifications, intersection of class specifications, clone.*

Д. Б. Буй, С. В. Компан

Діаграми класів ООП: формалізація та аналіз.

Дано порівняльний аналіз робіт, присвячених формальним моделям об'єктно-орієнтованого програмування (ООП). Предмет дослідження – модель діаграми класів (відповідна частково впорядкована множина). Модель класу (специфікації класу) – пара функціональних бінарних відношень, одна компонента уточнює атрибути, друга – методи. Успадкування уточнюється як включення графіків функцій. Над класами вводяться дві операції – перетин і зчленування. Формальні результати: структура частково впорядкованої множини класів, властивості операції накладення, в термінах якої вводиться операція зчленування: ідемпотентність, асоціативність, критерій комутативності. Модель діаграми класів можна використовувати для аналізу структури класів, що дозволяє виділяти підсистеми, клони та оптимізувати саму діаграму згідно з відповідним критерієм.

Ключові слова: *об'єктно-орієнтоване програмування (ООП), формальні моделі ООП, діаграма класів, операція зчленування класів, операція перетину класів, клон.*

Киевский национальный ун-т им. Тараса Шевченко
buy@unicyb.kiev.ua
robin_2005mail.ru

Получено 25.10.13