УДК 681.3.06

БОЛДАК А.А. ,
НЕВДАЩЕНКО М.В.

МАТЕМАТИЧЕСКИЙ АППАРАТ ФОРМАЛИЗАЦИИ МОДЕЛЕЙ, ИСПОЛЬЗУЕМЫХ ПРИ ПРОЕКТИРОВАНИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Формализованы понятия класса и отношений между классами, которые соответствуют терминам объектно-ориентированного подхода к проектированию информационных систем. Показано, что классы и операции, определенные на них, являются алгебраической системой, которая позволяет эффективно решать задачи, связанные с взаимнооднозначными преобразованиями моделей в рамках объектно-ориентированного подхода к проектированию информационных систем. Приведены примеры решения некоторых типичных задач с применением предложенного математического аппарата.

Classes and its relations concept that correspond the terms of the object-oriented information systems' design approach. Classes and its operations are the algebraic system. The system allows effective tasks' solving for model transformations in object-oriented design approach.

Проектирование программных систем связано с использованием различных моделей: концептуальных, проектных и моделей реализации.

Сложность задач, связанных с переходом от проектной модели к модели реализации, определяется необходимостью учета дополнительных ограничений, связанных с использованием выбранных программных технологий и инструментальных средств разработки. В случае использования многоуровневой архитектуры, ее решение становится еще более сложным, т.к. необходимо определить, каким образом проектные классы будут отображаться в одном или нескольких слоях такой архитектуры, тем более что на различных уровнях могут использоваться собственные формализмы для описания моделей.

Типичная в этом смысле ситуация возникает при реализации персистентных объектов, которые должны быть представлены как в программной среде, так и в СУБД.

Во многих источниках [1 – 5] описаны подходы и методы, которые могут применяться для решения подобных задач. Однако, по признанию самих авторов статей, область применимости предложенных ими методов ограничивается оптимизацией отдельных видов запросов [1, 3] и не может применяться для решения задачи отображения программных классов в СУБД в общем случае.

Целью работы является формулировка такого математического аппарата, который мог бы эффективно применяться для решении задач, свя-

занных с эквивалентными преобразованиями моделей в рамках объектно-ориентированного подхода к проектированию информационных систем.

В основе разрабатываемого математического аппарата лежит понятие класса, определяющего посредством спецификации типизированное множество и операции, которые образуют кольцо на множестве таких классов.

Алгебра Классов

Определение класса

Классом будем называть множество, определенное с помощью задания свойства

$$C = \{I \mid P(I)\}, \ (1)$$

где I – (от *Instance*) элемент (экземпляр) класса, в качестве которых могут выступать любые множества, в том числе и пустое, такие, что для них имеет место

$$\{i \in I \mid P(i)\} = \{i \mid P(i)\};$$

$P(I) = I \xrightarrow{P} \{0,1\}$ – (от *Property*) характеристическая функция, которая принимает значение 1, если I принадлежит классу, и 0 – в противном случае.

Характеристическая функция определяет наличие у элементов $i \in I$ некоторого свойства. Если экземпляры имеют несколько свойств, для каждой из которых можно определить $p_k, k = \overline{1,n}$, то общая характеристическая функция может быть представлена их конъюнкцией:

$$P(I) = \mathop{\&}_{k=1}^{n} p_k(I). \ (2)$$

Благодаря тому, что для логической операции конъюнкции имеет место коммутативность, функция $P(I)$ может взаимно однозначно отображаться во множество $\{p_1(I), p_2(I), ..., p_n(I)\}$, заданное перечислением свойств, которое будем называть спецификацией класса. Т.е.,

$$P(I) \leftrightarrow S = \{p_1(I), p_2(I), ..., p_n(I)\},$$

где S – (от *Specification*) спецификация класса.

Поскольку спецификация однозначно определяет характеристическую функцию класса, то она также определяет и класс. Класс, определенный с помощью спецификации, будем обозначать $C(S)$ или $C(S = \{p_1(I), p_2(I), ..., p_n(I)\})$.

Переход к спецификации класса, заданного перечислением свойств, дает возможность достаточно просто решать задачи классификации экземпляров.

По аналогии с классической теорией множеств определим понятие нуль-класса и универсального класса.

Нуль-классом является класс $C^0(S^0 = \varnothing)$.

Таким образом, если спецификация класса является пустым множеством, т.е. к элементам класса не выдвигается никаких требований, то любой экземпляр может рассматриваться как представитель этого класса.

Универсальным классом является класс $C^U(S^u = U_p)$, где S^u – универсальная спецификация, которая содержит множество всех свойств (универсум свойств).

Понятно, что U_p является конечным множеством, которое определяется в рамках каждой конкретной задачи. Однако, условие конечности U_p не нарушает условий универсума, т.е. всегда (в рамках задачи, которая рассматривается) имеет место

$$\forall S : S \cap S^U = S .$$

Смысл универсальной спецификации определяется тем, что к объекту в процессе его проектирования применяют макроподход [6], в рамках которого предполагается, что объект рассматривается как “черный ящик”, свойства которого определены и известны, но нет никаких предположений о его строении, что соответствует определению класса $C^U(S^U)$.

Суперклассом класса C есть класс $C_1(S_1) = SUPER[C(S)] : S_1 \subseteq S$, спецификация которого является подмножеством спецификации класса C.

Потомком класса C есть класс $C_1(S_1) = CHILD[C(S)] : S_1 \supseteq S$, если спецификация класса C является подмножеством спецификации класса C_1.

Очевидно, что отношение потомков и суперклассов являются обратными.

Операции с классами

Дополнения класса

Дополнением класса $C(S)$ до универсального класса $C^U(S^U)$ (или просто – дополнением класса $C(S)$) есть класс $\overline{C}(S^U \setminus S)$, определяемый спецификацией, которая является дополнением S до универсальной спецификации S^U.

Разностью классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \setminus C_2(S_1 \setminus S_2)$.

Обобщение классов

Обобщением классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \overset{G}{-} C_2(S_1 \cap S_2)$, определяемый спецификацией, которая получается в результате пересе-

чения спецификаций S_1 и S_2. Знак $\overset{G}{—}$ обозначает операцию обобщения (от *Generalization*).

Название операции отвечает процедуре определения общих (совместных) свойств объектов, в данном случае – классов $C_1(S_1)$ и $C_2(S_2)$. Очевидно, что множество этих свойств содержится в пересечении соответствующих спецификаций.

Расширение классов

<u>Расширением</u> классов $C_1(S_1)$ и $C_2(S_2)$ является класс $C_1 \overset{E}{—} C_2(S_1 \cup S_2)$, определяемый спецификацией, которую получают в результате объединения спецификаций S_1 и S_2. Знак $\overset{E}{—}$ обозначает операцию расширения (от *Extension*).

Свойства операции дополнения

Утверждение 1. Расширение класса и его дополнения образуют универсальный класс.

Действительно, если рассматривать класс $C(S)$ и класс $\overline{C(S)}$, то

$$C(S) \overset{E}{—} \overline{C(S)} = C(S \cup \overline{S}) = C(S^U) = C^U .$$

Утверждение 2. Обобщение класса и его дополнения образуют нуль-класс.

По аналогии с доказательством предыдущего утверждения

$$C(S) \overset{G}{—} \overline{C(S)} = C(S \cap \overline{S}) = C(\varnothing = S^0) = C^0 .$$

Утверждение 3. Дополнение к дополнению класса является самим классом.

Т.е., $\overline{\overline{C(S)}} = \overline{C(\overline{S})} = C(\overline{\overline{S}}) = C(S)$.

Утверждение 4. Для классов имеет место соотношение $C_1 \setminus C_2 = C_1 \overset{G}{—} \overline{C_2}$.

Учитывая свойства обобщения, получим

$$C_1(S_1) \overset{G}{—} \overline{C_2(S_2)} = C(S_1 \cap \overline{S_2}) = C(S_1 \cap S^u \setminus S_2) = C((S_1 \cap S^u) \setminus S_2) =$$

$$C(S_1 \setminus S_2) = C(S_1) \setminus C(S_2).$$

Свойства операции обобщения

Утверждение 5. Обобщение классов является коммутативным $C_1 \overset{G}{—} C_2 = C_2 \overset{G}{—} C_1$.

Справедливость этого утверждения следует из коммутативности операции пересечения множеств.

$$C_1(S_1) \overset{G}{—} C_2(S_2) = C(S_1 \cap S_2) = C(S_2 \cap S_1) = C_2 \overset{G}{—} C_1$$

Утверждение 6. Обобщение классов является суперклассом для всех его потомков.

Основываясь на определении суперкласса и операции обобщения, получим

$$C_1(S_1) \overset{G}{—} C_2(S_2) = C(S = S_1 \cap S_2), S \subseteq S_1, S \subseteq S_2 \Rightarrow$$

$$C_1(S_1) \overset{G}{—} C_2(S_2) = SUPER(C_1), C_1 = CHILD(C_1(S_1) \overset{G}{—} C_2(S_2));$$

$$C_1(S_1) \overset{G}{—} C_2(S_2) = SUPER(C_2), C_2 = CHILD(C_1(S_1) \overset{G}{—} C_2(S_2)) .$$

Утверждение 7. Обобщение класса с собою является самим классом $C \overset{G}{—} C = C$.

Утверждение 8. Обобщение класса и нуль-класса является нуль-классом $C \overset{G}{—} C^0 = C^0$.

Доказательства утверждений 7 и 8 тривиальны.

Свойства операции расширения

Утверждение 9. Расширение классов является коммутативным $C_1 \overset{E}{—} C_2 = C_2 \overset{E}{—} C_1$.

По аналогии с доказательством утверждения 5 получаем

$$C_1(S_1) \overset{E}{—} C_2(S_2) = C(S_1 \cup S_2) = C(S_2 \cup S_1) = C_2 \overset{E}{—} C_1$$

Утверждение 10. Любой класс является суперклассом для его расширения с любым другим классом.

Действительно,

$$C_1(S_1)\overset{E}{—}C_2(S_2) = C(S = S_1 \cup S_2), S_1 \subseteq S, S_2 \subseteq S \Rightarrow$$

$$C_1(S_1)\overset{E}{—}C_2(S_2) = CHILD(C_1), C_1 = SUPER(C_1(S_1)\overset{E}{—}C_2(S_2));$$

$$C_1(S_1)\overset{E}{—}C_2(S_2) = CHILD(C_2), C_2 = SUPER(C_1(S_1)\overset{E}{—}C_2(S_2)).$$

Утверждение 11. Расширение класса с собой – есть сам класс $C\overset{E}{—}C = C$.

Утверждение 12. Расширение класса и нуль-класса – есть сам класс $C\overset{E}{—}C^0 = C$.

Связь прямого произведения множеств с операцией расширения классов

Известно, что операция прямого произведения множеств является единственным средством создания агрегатов, т.е. множеств, элементы которых состоят из нескольких частей. Рассмотрим, как влияет операция прямого произведения множеств на классы, с помощью которых эти множества задаются.

Допустим, что есть множества $A = \{a \mid \underset{S_A}{\&}\, p(a)\}$ и $B = \{b \mid \underset{S_B}{\&}\, r(b)\}$, которые описываются соответствующими классами $C_A(S_A = \{p(A)\})$, $C_b(S_b = \{r(B)\})$. Аргументами функций $p(a)$ и $r(b)$ являются, соответственно, элементы множеств *A* и *B*.

Вследствие прямого произведения $A \times B$ получаются элементы ⟨*a*,*b*⟩, для которых одновременно должны выполнятся ограничения $\underset{S_A}{\&}\, p(a)$ и $\underset{S_B}{\&}\, r(b)$, наложенные отдельно на множества *A* и *B*, т.е. $S_{A\times B} = \underset{S_A}{\&}\, p(a) \,\&\, \underset{S_B}{\&}\, r(b)$, но в этом случае функции *p* и *r* своими аргументами имеют кортежи ⟨*a*,*b*⟩. Приведение функций *p* и *r* к общей области определения выполняется с помощью проекций

$$S_{A\times B} = \underset{S_A}{\&}\, p(\Pi_A\langle a,b\rangle) \,\&\, \underset{S_B}{\&}\, r(\Pi_b\langle a,b\rangle).$$

Учитывая то, что $p(\Pi_A\langle a,b\rangle) = \Pi_A p(\langle a,b\rangle) = p'(\langle a,b\rangle)$ – некоторая функция, определенная на множестве $A \times B$ (как и для r), предыдущую формулу можно переписать в виде

$$S_{A\times B} = \mathop{\&}_{S_A} p(\langle a,b\rangle) \,\&\, \mathop{\&}_{S_B} r(\langle a,b\rangle)\,.$$

Учитывая ассоциативность и коммутативность операции конъюнкции, можно утверждать, что

$$C_{A\times B} = C_A \overset{E}{—} C_B\,.$$

Последнее имеет важное практическое значение, поскольку позволяет упрощенно (по отношению к операции прямого произведения) определять агрегатные объекты с помощью операции расширения, в основе которой лежит объединение соответствующих конечных спецификаций, заданных набором свойств.

Определение алгебры классов

Утверждение 13. Операции расширения, обобщения и дополнения образуют, по крайней мере, кольцо на множестве классов.

Для доказательства этого утверждения необходимо проверить ограничения, которым должна удовлетворять рассматриваемая система.

<u>Ассоциативность</u> операции расширения классов следует из свойств спецификаций, а именно

$$C_1 \overset{E}{—} (C_2 \overset{E}{—} C_3) = C_1 \overset{E}{—} C(S_2 \cup S_3) = C(S_1 \cup (S_2 \cup S_3)) = C((S_1 \cup S_2) \cup S_3) =$$

$$C(S_1 \cup S_2) \overset{E}{—} C_3 = (C_1 \overset{E}{—} C_2) \overset{E}{—} C_3.$$

<u>Единицей</u> для операции расширения является ноль-класс C^0 (см. Утверждение 12).

<u>Противоположным элементом</u> класса C по операции расширения является его дополнение $\overline{C}$ (см. Утверждение 2).

Операция расширения является <u>коммутативной</u> (см. Утверждение 9).

Таким образом, для множества классов и операции расширения выполняются условия существования группы, т.е. $(\overset{E}{—}, \{C\}, C^0)$ является абелевой группой.

Для операции обобщения очевидна <u>замкнутость</u>, <u>ассоциативность</u> доказывается аналогично предыдущему.

Установлено, что обобщение является коммутативной операцией (см. Утверждения 5), т.е. левый и правый дистрибутивные законы объединяются.

Покажем, что для расширения и обобщения имеет место дистрибутивный закон.

$$C_1(S_1)\overset{G}{-}[C_2(S_2)\overset{E}{-}C_3(S_3)] = C_1(S_1)\overset{G}{-}C(S_2 \cup S_3) = C(S_1 \cap (S_2 \cup S_3)) =$$

$$C((S_1 \cap S_2) \cup (S_1 \cap S_3)) = C(S_1 \cap S_2)\overset{E}{-}C(S_1 \cap S_3) = \qquad (3)$$

$$[C_1(S_1)\overset{G}{-}C_2(S_2)]\overset{E}{-}[C_1(S_1)\overset{G}{-}C_3(S_3)]$$

Таким образом, выполняются также требования, обеспечивающие существования кольца, т.е. $(\overset{G}{-},\overset{E}{-},\{C\},C^0,C^U)$ является коммутативным кольцом с единицей C^U.

Преобразования диаграмм классов

Приведенная выше теория классов, по своей терминологии совпадает с терминами объектно-ориентированного проектирования, которое в настоящее время используется в качестве методологической основы при проектировании информационного и программного обеспечения [7].

Выводы о том, что $(\overset{G}{-},\overset{E}{-},\{C\},C^0,C^U)$ является замкнутой алгебраической системой, позволяет получить преобразования классов и их взаимоотношений, которые, с одной стороны, сохраняют свойства классов, с другой – учитывают дополнительные ограничения.

В качестве графического представления классов и отношения между ними будем использовать UML–диаграммы [8]. Пример такой диаграммы и необходимые объяснения изображены на Рис. 1.

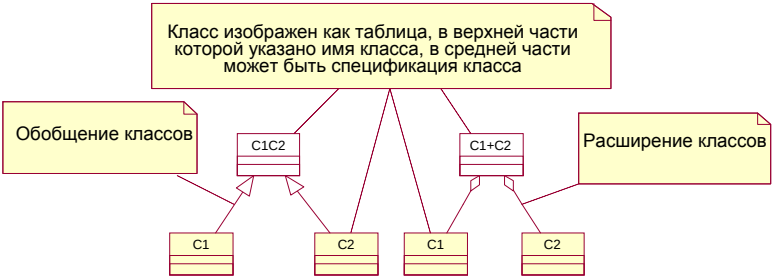

Рис. 1. Пример UML–диаграммы

Взаимное преобразование обобщения и расширения классов

Необходимость таких преобразований обусловлена тем, что в некоторых моделях, а, соответственно, и в языках программирования, не используется понятие обобщений, например в реляционной модели данных [9], которая поддерживается большинством СУБД.

В этом случае необходимы преобразования обобщений классов, т.е. такое эквивалентное представление, в котором сохраняются свойства классов и отсутствуют их обобщения.

Утверждение 14. Любой класс может быть представлен расширением своего суперкласса и его дополнения к рассматриваемому классу.

Пусть рассматривается класс *C*, при чем известно, что $C = CHILD(C_S)$. Тогда:

$$C_S \overset{E}{—} (C \setminus C_S) = C_S \overset{E}{—} (C \overset{G}{—} \overline{C_S}) \quad (4)$$

Учитывая то, что $S \supseteq S_S$, имеет место поглощение $C_S \overset{G}{—} C = C_S$.

Таким образом, использование поглощения, дистрибутивного закона и свойств обобщения в (4) дает следующий результат:

$$C_S \overset{E}{—} (C \overset{G}{—} \overline{C_S}) = (C \overset{G}{—} C_S) \overset{E}{—} (C \overset{G}{—} \overline{C_S}) = C \overset{G}{—} (C_S \overset{E}{—} \overline{C_S}) = C \overset{G}{—} C^U = C$$

Изобразим этот вывод в виде UML– диаграммы (Рис. 2).

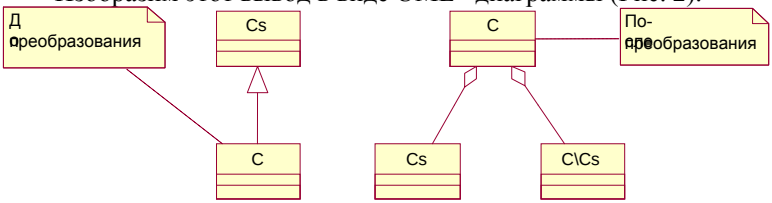

Рис. 2. Взаимные преобразования обобщения и расширения классов

Приведем пример использования эквивалентных преобразований для обобщения и расширения.

При разработке базы данных для учета успеваемости учеников были определены классы УЧЕНИК ({ФИО, Адрес, Класс}) и УЧИТЕЛЬ ({ФИО, Адрес, Должность}), на которых можно получить обобщение ПЕРСОНА ({ФИО, Адрес}). Соответствующая диаграмма изображена на Рис. 3.

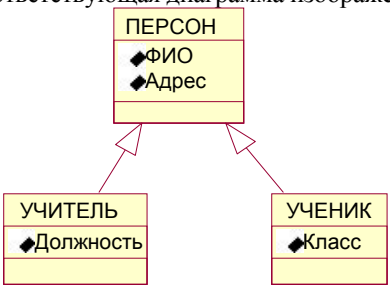

Рис. 3. Диаграмма классов до преобразования

Однако, для того, чтобы определить эти классы в реляционной модели необходимо избавится от обобщений, но таким образом, чтобы класс ПЕРСОНА сохранился. Это возможно благодаря утверждению 4 (Рис. 4).

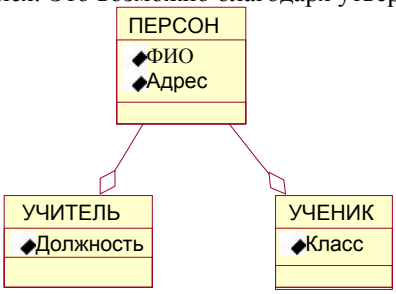

Рис. 4. Диаграмма классов после преобразования

Таким образом, в соответствии с реляционной моделью, этот фрагмент можно представить тремя таблицами, которые связаны между собой внешними ключами. Внешние ключи реализуют в данном случае операцию расширения.

Соответственно, на языке SQL эта диаграмма описывается следующим образом:

```
CREATE TABLE ПЕРСОНА    (     Id NUMBER PRIMARY KEY,
                              ФИО TEXT,
                              Адрес TEXT
                        );
CREATE TABLE УЧЕНИК     (     Id NUMBER PRIMARY KEY,
                              Класс TEXT,
                              Персона          NUMBER
                              REFERENCES ПЕРСОНА(Id)
                        );
CREATE TABLE УЧИТЕЛЬ    (     Id NUMBER PRIMARY KEY,
                              Должность TEXT,
                              Персона          NUMBER
                              REFERENCES ПЕРСОНА(Id)
                        );
```

Разрешение проблемы множественного наследования

При построении проектной модели на множестве классов могут выполняться обобщения, в том числе и множественные.

С другой стороны, большинство объектно-ориентированых языков программирования реализуют только простое наследование. Следовательно, необходимы преобразования, которые позволят избавиться от множественного наследования при условии сохранения свойств классов.

Непосредственное использование дистрибутивного закона алгебры классов позволяет решить эту задачу.

Начальная диаграмма изображена на Рис. 5, на котором класс C_1 имеет два суперкласса.

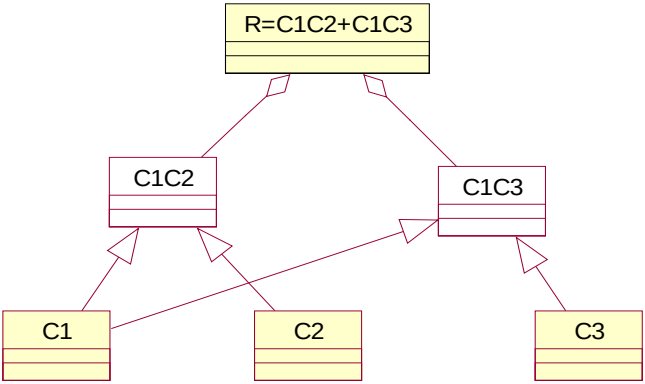

Рис. 5. Пример множественного наследования

Применение дистрибутивного закона позволяет решить эту задачу так, как это показано на Рис. 6.

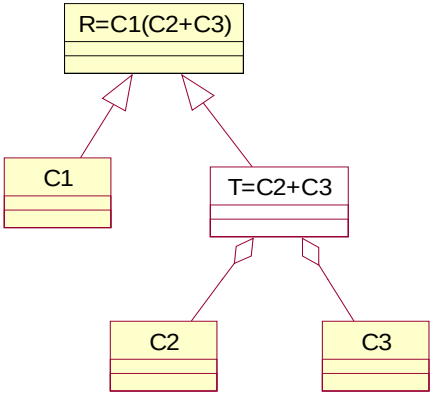

Рис. 6. Решение задачи исключения множественного наследования

Используя модель, показанную на Рис. 6 в языке Object Pascal [9], классы могут быть описаны следующим образом:

```
Type
R = class
…
end;
```

```
C1 = class( R )
…
end;
C2 = class
…
end;
C3 = class
…
end;
T = class( R )
InstanceC2 : C2;
InstanceC3 : C3;
…
end;
```

Следует отметить, что для схема, изображенная на Рис. 5 в языке Object Pascal вообще не может быть описана.

Наследование расширения классов

При проектировании структуры классов часто обнаруживается подобность агрегатных отношений. В контексте обобщения структур интересен ответ на вопрос: являются ли агрегатные отношения свойствами и могут ли они наследоваться.

Рассмотрим диаграмму классов, показанную на Рис. 7.

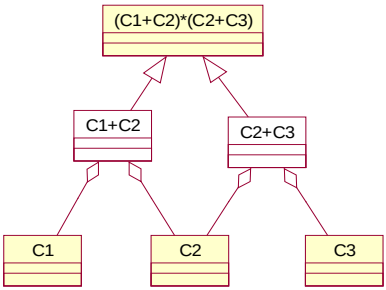

Рис. 7. Пример однотипных расширений.

Как видно на рисунке, расширение с классом C_2 является общим свойством для C_1+C_2 и C_2+C_3.

Применив преобразования (3), (Утверждение 7), (Утверждение 9), (Утверждение 11) и (Утверждение 13), получим

$$(C_1 \overset{E}{-} C_2) \overset{G}{-} (C_2 \overset{E}{-} C_3) = (C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \overset{E}{-} (C_2 \overset{G}{-} C_2) \overset{G}{-} (C_2 \overset{E}{-} C_3) =$$

$$\left[(C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \right] \overset{E}{-} \left[C_2 \overset{G}{-} (C_2 \overset{E}{-} C_3) \right] =$$

$$(C_1 \overset{G}{-} C_2) \overset{E}{-} (C_1 \overset{G}{-} C_3) \overset{E}{-} C_2 = \left[C_2 \overset{E}{-} (C_1 \overset{G}{-} C_2) \right] \overset{E}{-} (C_1 \overset{G}{-} C_3) =$$

$$C_2 \overset{E}{-} (C_1 \overset{G}{-} C_3).$$

На Рис. 8 показан результат.

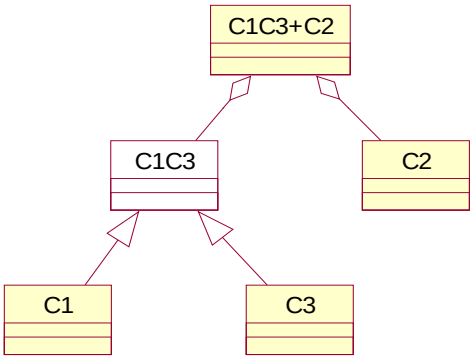

Рис. 8. Наследование общих расширений

Таким образом, расширения могут рассматриваться как особенные свойства (в объектно-ориентированном программировании их называют агрегатными) и они могут наследоваться, как и все остальные.

Выводы

Разработанный математический аппарат, представляющий собой алгебру, определенную на множестве типизированных классов, может использоваться для решения задач, возникающих в процессе проектирования информационных систем и связанных с взаимнооднозначными преобразованиями моделей. Его формальность создает предпосылки для автоматизации решения таких задач на ранних стадиях проектирования.

Список использованной литературы

1. Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke, ajmohan Rajaraman. Multi-Query Optimization for Sensor Networks. International Conference on Distributed Computing in Sensor Systems, 2005.

2. Dunren Che, Karl Aberer. A Heuristics-Based Approach to Query Optimization in Structured Document Databases. IEEE Computer Society, 1999.

3. Vijay M. Sarathy, Lawrence V. Saxton, Dirk Van Gucht. An Object Based Algebra for Parallel Query Processing and Optimization. Technical Report TR368, Dept. of Computer Science, Indiana University, December 1992.

4. Nick Taylor. Chaudhuri: An Overview of Query Optimization in Relational Systems. University of Pennsylvania, October 2005.

5. Jef Wijsen, Alexis B`es. On Query Optimization in a Temporal SPC Algebra. Elsevier, 2003.

6. Бузовский О.В. Формальное определение понятий объекта и процесса проектирования.: Весник КПИ. Серия «Автоматика и электроприборостроение», №29: К.,1993.

7. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер.с анл. – М.:Конрод, 1992.

8. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя.: Пер.с анл. – М.:ДМК, 2000.

9. Дейт К. Дж. Введение в системы баз данных, 6-е издание: Пер.с анл. – К.;М.;СПб: Издательский дом «Вильямс», 1999.