ИНФОРМАТИКА И УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ И СОЦИАЛЬНЫХ СИСТЕМАХ

УДК 004.031 ***DOI: 10.46960/1816-210X_2022_2_7***

МОДЕЛЬ И ЛИНГВИСТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ НИЗКОУРОВНЕВОЙ СТРУКТУРНОЙ МОДИФИКАЦИИ ОБЪЕКТНО-ОРИЕНТИРОВАННЫХ СИСТЕМ

Д.В. Жевнерчук
ORCID: **0000-0001-6306-0893** e-mail: **zhevnerchuk@yandex.ru**
Нижегородский государственный технический университет имени Р.Е. Алексеева
Нижний Новгород, Россия

А.С. Захаров
ORCID: **0000-0002-4058-0746** e-mail: **lukalex.nnov@gmail.com**
Нижегородский государственный технический университет имени Р.Е. Алексеева
Нижний Новгород, Россия

Анализируется проблема применения генераторов программных компонентов, автоматизирующего ряд задач по построению объектно-ориентированных программных систем, но оставляющего в то же время степень участия человека-эксперта, архитектора, разработчика в данном процессе достаточно высокой. Приведены результаты исследований, направленных на развитие принципов *Inversion of Control* и средств интеллектуальной поддержки структурной организации объектно-ориентированных систем, преодолевающих ограничения одного из принципов *SOLID*, а именно «open-close». Рассматривается процесс модификации объектно-ориентированных систем (ООС) в контексте структурных изменений, не нарушающих низкоуровневые правила построения ООС. Такой процесс является базовым для реконфигурирования, реструктуризации и рефакторинга ООС в условиях меняющихся внешних факторов, а также требований, предъявляемых к системе. Показано, что модификация любой ООС может быть представлена с помощью дизъюнкта Хорна. Обсуждаются результаты эксперимента по низкоуровневой структурной модификации UI-систем.

Ключевые слова: объектно-ориентированное проектирование, UML, открытая информационная система, модификация, дизъюнкт Хорна.

ДЛЯ ЦИТИРОВАНИЯ: Жевнерчук, Д.В. Модель и лингвистическое обеспечение низкоуровневой структурной модификации объектно-ориентированных систем / Д.В. Жевнерчук, А.С. Захаров // Труды НГТУ им. Р.Е. Алексеева. 2022. № 2. С. 7-16. DOI: 10.46960/1816-210X_2022_2_7

MODEL AND LINGUISTIC SUPPORT OF LOW-LEVEL STRUCTURAL MODIFICATION OF OBJECT-ORIENTED SYSTEMS

D.V. Zhevnerchuk
ORCID: **0000-0001-6306-0893** e-mail: **zhevnerchuk@yandex.ru**
Nizhny Novgorod State Technical University n.a. R.E. Alekseev
Nizhny Novgorod, Russia

A.S. Zakharov
ORCID: **0000-0002-4058-0746** e-mail: **lukalex.nnov@gmail.com**
Nizhny Novgorod State Technical University n.a. R.E. Alekseev
Nizhny Novgorod, Russia

Abstract. The use of software components generators that automate a number of tasks for the construction of object-oriented software systems, but at the same time with a quite high degree of participation of a human expert, architect, developer in this process, is analyzed. Results of research aimed at developing of the Inversion of Control principles and intellectual support means for the structural organization of object-oriented systems that overcome the limitations of one of the SOLID principles, namely «open-close», are presented. Process of modification of object-oriented systems (OOS) is considered in the context of structural changes that do not breach the low-level rules of OOS construction. Such a process is the basic one for reconfiguring, restructuring and refactoring of OOS in conditions of changing external factors, as well as the system requirements. It is shown that modification of any OOS can be presented using the Horn clause. Results of the experiment on low-level structural modification of UI systems are discussed.

Key words: object-oriented design, UML, open information system, modification, the Horn clause.

FOR CITATION: D.V. Zhevnerchuk, A.S. Zakharov. Model and linguistic support of low-level structural modification of object-oriented systems. Transactions of NNSTU n.a. R.E. Alekseev. 2022. № 2. Pp. 7-16.
DOI: 10.46960/1816-210X_2022_2_7

Введение

Существует множество различных определений понятия модифицируемости [1]. Согласно общепринятой формулировке, модифицируемость (modifiability) как свойство системы определяет степень простоты эффективного и рационального изменения продукта или системы без добавления дефектов и снижения качества продукта [1]. Будем считать, что объектно-ориентированные системы (ООС) обладают свойством модифицируемости, если структурные изменения, получаемые в результате применения комбинаций допустимых операций пространственного и семантического сопряжения блоков, а также удаления блоков из многоблочных структур, не нарушают их целостности в контексте спецификации UML. С учетом этого о модифицируемости можно говорить в контексте структурных изменений ООС, не нарушающих базовые (низкоуровневые) правила построения ООС, при этом процесс модификации объектно-ориентированных систем является иерархическим (рис. 1).

Процессы низкоуровневой структурной модификации объектно-ориентированных систем выступают фундаментом для реконфигурирования и рефакторинга ООС в условиях требований и ограничений различной природы, а также для создания самоорганизующихся и адаптивных ООС.

Теоретическая часть

Любая объектно-ориентированная система может быть представлена представима совокупностью множеств блоков *(B)* двух типов и связей*(L)*двух типов между ними:

$$S = < B^P, B^D, L^U, L^E >, \quad (1)$$

где:

B^P – множество блоков моделей фрагментов предметной области или информационных процессов;

L^U – множество семантических связей, регламентируемых подмножеством языка моделирования систем UML;

L^E – множество связей, определяющих пространственную ориентацию блоков друг относительно друга после их сопряжения.

Таким образом, любой блок ООС синтезируется на основе согласованной совокупности спецификаций или подмножества одной спецификации, ориентированной на решение определенной задачи (реализацию заданной функции либо группы функций) [2]. Множество B^P содержит части целевой ООС, необходимые для описания предметных областей или информационных процессов:

- абстрактные и конкретные типы;
- атрибуты и методы;
- экземпляры, порождаемые от конкретных типов, которые подвергаются обработке;

- модули, группирующие типы.

Представим B^P в виде кортежа, элементами которого являются множества блоков, инкапсулирующих выделенные части ООС:

$$B^P =< C, B^a, B^m, O, M >, \quad (2)$$

где:
$C =< C^a, C^c >$ – множество типов (C^a – абстрактный тип, C^c – конкретный тип, от которого можно создавать экземпляры;
B^a – множество блоков атрибутов;
B^m – множество блоков методов;
O – множество экземпляров (объектов);
M – множество модулей ООС.

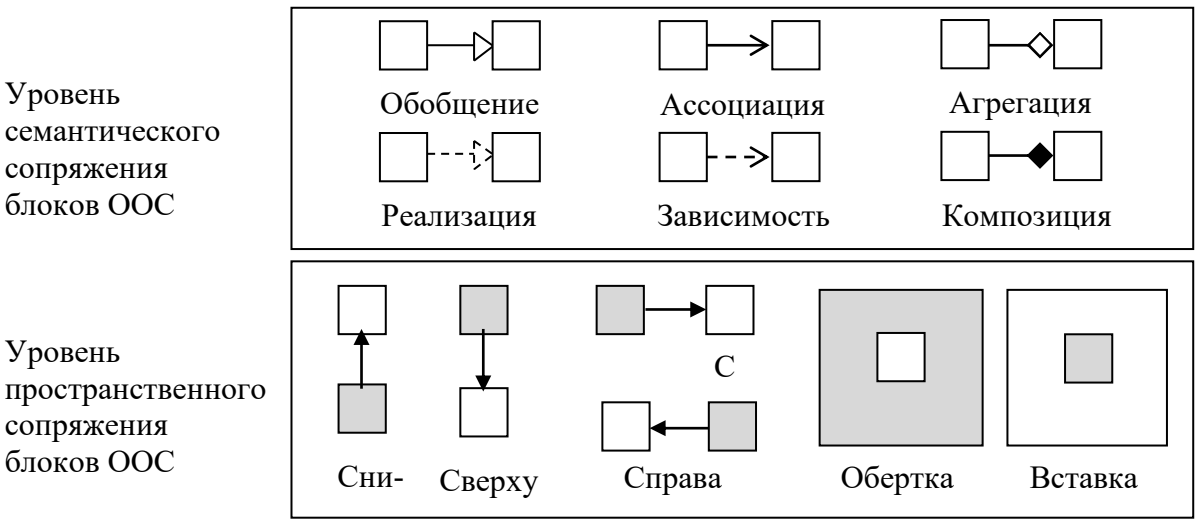


Рис. 1. Обобщенная схема низкоуровневой структурной модификации объектно-ориентированной системы

Fig. 1. Object-oriented system low-level structural modification generalized scheme

Элементами множества B^D являются формализованные стандартизированные профили ООС, в том числе, сигнатуры, определяющие режимы сопряжения блоков, являющихся элементами множества B^P:

- сигнатура абстрактного / конкретного типа;
- сигнатура метода;
- сигнатура модуля.

Представим B^D в виде кортежа, элементами которого являются множества блоков, инкапсулирующих сигнатуры выделенных частей ООС:

$$B^D =< D^{C^a}, D^{C^c}, D^{B^m}, D^M >, \quad (3)$$

где: D^{C^a} – сигнатура абстрактного типа;
D^{C^c} – сигнатура конкретного типа;
D^{B^m} – сигнатура метода;
D^M – сигнатура модуля.

Элементами множества L^E являются связи, формализующие операции пространственного сопряжения блоков ООС:

1) сопряжение блоков снизу $b_i \frown b_j \in B$;
2) сопряжение блоков справа $b_i \frown b_j \in B$;
3) сопряжение блоков сверху $b_i \frown b_j \in B$;
4) сопряжение блоков слева $b_i \frown b_j \in B$;
5) сопряжение типа «вставка» $b_i \frown b_j \in B$;
6) сопряжение типа «обертка» $b_i \frown b_j \in B$;
7) сопряжение, в результате которого первый блок-операнд оказывается поверх второго блок-операнда $b_i \frown b_j \in B$;
8) сопряжение, в результате которого первый блок-операнд оказывается за вторым блок-операндом $b_i \frown b_j \in B$.

Введем следующие обозначения для аналитического описания семантических связей, регламентируемых подмножеством языка UML и являющихся элементами множества L^U:

1) обобщение $b_i \triangleright b_j \in B$;
2) реализация $b_i \dashrightarrow b_j \in B$;
3) ассоциация $b_i > b_j \in B$;
4) зависимость $b_i \dashrightarrow b_j \in B$;
5) агрегация $b_i \diamond b_j \in B$;
6) композиция $b_i \blacklozenge b_j \in B$.

Дополним множество L^U связями инстанцирования и поведения:

1) инстанцирование $b_i \square b_j \in B$, $b_i \in C^c$, $b_j \in O$;
2) поведение $b_i.b_j \in B$, $b_i \in O$, $b_j \in B^m$.

Поведение объекта может быть выражено через инстанцирование и сопряжение типа «Вставка». Из определения инстанцирования [3] следует:

$$\forall b_i \in C^c, b_j \in O,\ b_k \in B^m : (b_i \square b_j) \cap (b_k \frown b_i) \rightarrow (b_j . b_k) \qquad (4)$$

Выполним формализацию паттерна «Делегат», который является базовым паттерном объектно-ориентированного проектирования. Из определения делегирования [3] следует:

$$\begin{gathered} \exists b_h \in B^H,\ b_{i,j} \in C^c,\ b_{k,l} \in O,\ b_{q,r} \in B^m,\ b_t \in B^T,\ b_a \in B^A : \\ (b_t \frown b_h) \cap (b_h \frown b_j) \cap (b_l \square b_j) \cap (b_q \frown b_i) \cap (b_r \frown b_j) \cap \\ (b_t \frown b_a) \cap (b_a \frown b_i) \cap ((b_l . b_r) \frown b_q) \rightarrow (b_i \Theta b_j) \end{gathered} \qquad (5)$$

Согласно (5), существуют:

- сигнатура b_h, в которой разрешен тип b_t;
- сигнатура b_h определена для класса b_j;
- экземпляр b_l класса b_j;
- методы b_q, b_r определены в составе классов соответственно b_i, b_j;
- тип b_t определен в секции атрибутов b_a;
- секция атрибутов b_a определена в составе класса b_i;
- вызов метода b_r у экземпляра b_l определен в составе метода b_q.

Совокупность фактов по выражению (5) определяет факт делегирования поведения классом b_i классу b_j. Для каждого элемента из множества L^U существует единственная стрелка диаграммы классов UML, начало которой связано с первым блоком-операндом, а конец – со вторым. Операции сопряжения блоков и операции, определяющие связи, регламентированные UML представляют разновидности более общей операции сопряжения блоков [4,5]. Поскольку операция сопряжения на множестве блоков представляет собой алгебраическую группу, b_i может представлять собой как многоблочную структуру, так и элементарный (неделимый) блок.

Для обозначения обобщенной операции удаления блока b_i из многоблочной структуры b_j введем следующее обозначение:

$$b_i / b_j \in B \quad (6)$$

Утверждение 1. *Процесс модификации ООС может быть представлен комбинацией CRUD-операций (акроним от Create, Read, Update, Delete) над кортежем (1)*

■

В самом общем случае любая объектно-ориентированная система может быть представлена блочно-иерархической структурой [3]. На каждом уровне ее элемент представляет собой компонент, инкапсулирующий свойства и поведение и обладающий интерфейсами для сопряжения с внешними компонентами. В свою очередь, каждый компонент может быть декомпозирован и представлен многокомпонентной структурой следующего уровня. Листья иерархии содержат описание неделимых элементов или компонентов, декомпозиция которых не имеет смысла в условиях выбранного уровня абстракции. На нижнем уровне любая объектно-ориентированная система может быть представлена многокомпонентной структурой на основе операций пространственного и семантического сопряжения, причем операции семантического сопряжения трактуются в контексте модели классов спецификации UML.

Согласно концепции технической самоорганизации открытых информационных систем [5-7], должен существовать информационный процесс *I* между ООС и внешней средой, способный переводить ООС между равновесными состояниями, в которых ООС удовлетворяет ограничениям пространственного и семантического сопряжения ее элементов. Необходимым условием обеспечения информационного процесса *I* является существование во внешней среде единого стандартизированного информационного пространства (ЕСИП), являющегося источником компонентов, которые должны быть выбраны и встроены в ООС в процессе ее модификации.

Таким образом, любая операция модификации ООС является частным случаем операции Update (U) и представляет собой комбинацию следующих операций:

- поиск утилизируемых компонентов (многокомпонентных структур) ООС, подлежащих замене;
- поиск компонентов (многокомпонентных структур) в ЕСИП, которые могут быть встроены в ООС вместо утилизируемых компонентов без нарушения правил пространственного и семантического сопряжения;
- удаление утилизируемых компонентов (многокомпонентных структур) из ООС;
- вставка новых компонентов (многокомпонентных структур), полученных из ЕСИП в структуру ООС.

Очевидно, что выделенные операции являются частными случаями операций Create, Read, Delete.

■

Следствие 1. Существует внешняя управляющая система, выполняющая отображение множества компонентов ООС на множество компонентов ЕСИП с сохранением правил пространственного и семантического сопряжения.

■

Выше было показано, что между множеством компонентов ООС и ЕСИП существует отображение. Поскольку ООС и ЕСИП являются открытыми информационными системами, они могут быть представлены в виде компонентов, обладающих стандартизированными интерфейсами. Таким образом, возможны три случая размещения системы управления отображением между ООС и ЕСИП (далее, система управления отображением − СУО).

1. СУО встраивается в ООС и представляет собой отдельный модуль, являющийся частью изменяемой ООС. Недостатками этого варианта является сильная связанность ООС с модулями СУО, а также отсутствие, в общем случае, конфигурируемой встраиваемой СУО.

2. СУО встраивается в ЕСИП и представляет собой отдельный модуль, являющийся его частью. Недостатки этого варианта совпадают с недостатками встраивания ЕСИП в ООС. Кроме того, система сильно усложняется с точки зрения проектирования и сопровождения.
3. СУО является самостоятельной внешней системой. Последний вариант лишен недостатков первого и второго вариантов, но при организации взаимодействия СУО с ООС и ЕСИП требует развитой инфраструктуры и более сложного API.

Таким образом, существование внешней управляющей системы обосновано.

■

Утверждение 2. *Модификация любой ООС может быть формализована с помощью дизъюнкта Хорна.*

■

ООС, не содержащую ни одного блока, будем обозначать значком пустого множества ∅. Очевидно, что операция удаления блока из одноблочной ООС вернет ∅: $b_i / b_i = \varnothing$

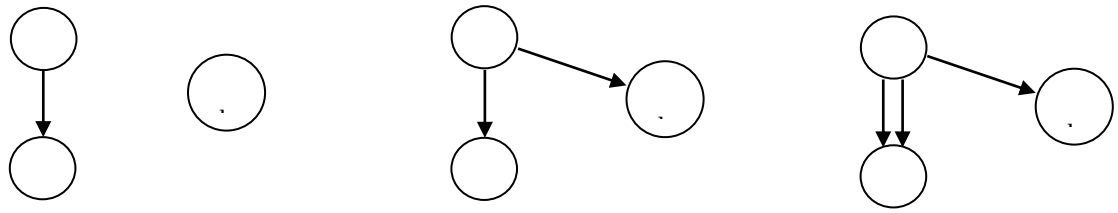
Рассмотрим применение операции удаления блока из двухблочных ООС.

1. Блоки $b_i \in C^c$, $b_j \in C$ связаны отношением обобщения/зависимости:
   $(b_i \triangleright b_j) / b_i = b_j$ удаление дочернего блока
   $(b_i \triangleright b_j) / b_i = b_j$ удаление зависимого блока
   $\forall b_i$: $(b_i \triangleright b_j) / b_j = \varnothing$ удаление родительского блока
   $\forall b_i$: $(b_i \triangleright b_j) / b_j = \varnothing$ удаление блока, от которого зависят другие блоки
2. Блоки $b_i \in C^c$, $b_j \in C$ связаны отношением ассоциации/агрегации
   Удаление блока агрегата:
   $(b_i \diamond b_j) / b_j = b_i$
   $(b_i \diamond b_j) / b_j = b_i$
   Удаление блока, который определяет часть агрегата (для случая ассоциации выражение будет таким же):
   $(b_i \diamond b_j) / b_i \Rightarrow \exists b_k \in B^H, b_l \in B^T, b_r \in B^A$:
   $[(b_l \bigcirc b_k) \cap (b_l \bigcirc b_r) \cap (b_k \bigcirc b_i) \cap (b_r \bigcirc b_j) \cap (b_i \diamond b_j)] / b_i = b_j / b_r$
3. Блоки $b_i \in C^c$, $b_j \in C$ связаны отношением композиции
   Удаление блока композита:
   $(b_i \blacklozenge b_j) / b_j = \varnothing$
   Удаление блока, который определяет часть композита:
   $(b_i \blacklozenge b_j) / b_i \Rightarrow \exists b_k \in B^H, b_l \in B^T, b_r \in B^A$:
   $[(b_l \bigcirc b_k) \cap (b_l \bigcirc b_r) \cap (b_k \bigcirc b_i) \cap (b_r \bigcirc b_j) \cap (b_i \blacklozenge b_j)] / b_i = b_j / b_r$
4. Блоки $b_i \in C^c$, $b_j \in C^a$ связаны отношением реализации
   Удаление блока, реализующего спецификацию:
   $\forall b_i, b_j$ **:** $(b_i \triangleright b_j) / b_i = b_j$
   Удаление блока-спецификации:
   $\forall b_i, b_j$ **:** $(b_i \triangleright b_j) / b_j \Rightarrow \exists b_k \in B^M$:
   $[(b_k \bigcirc b_i) \cap (b_k \bigcirc b_j) \cap (b_i \triangleright b_j)] / b_j = b_i / b_k$

Рассмотрим операции удаления блоков и связей между блоками из n-блочных ООС. Введем дополнительно концепт «мусорная корзина» (trashbin или tb), а также дополнительную связь, формализующую семантику удаления блоков ООС. На рис. 2а представлена связка блоков «a» и «b» и экземпляр «tb». При удалении блока «a» построим связь «a» с «tb» (рис. 2б). Альтернативным вариантом является добавление рефлексивной связи, формализующей процесс удаления блока (рис. 2в).

Удаление блока может сопровождаться удалением входящих/исходящих связей его с другими блоками. Рассмотрим случай удаления связи между блоками «a» и «b» (рис. 3а). Введем дополнительно концепты, экземпляры «l» которых формализуют тип связи между

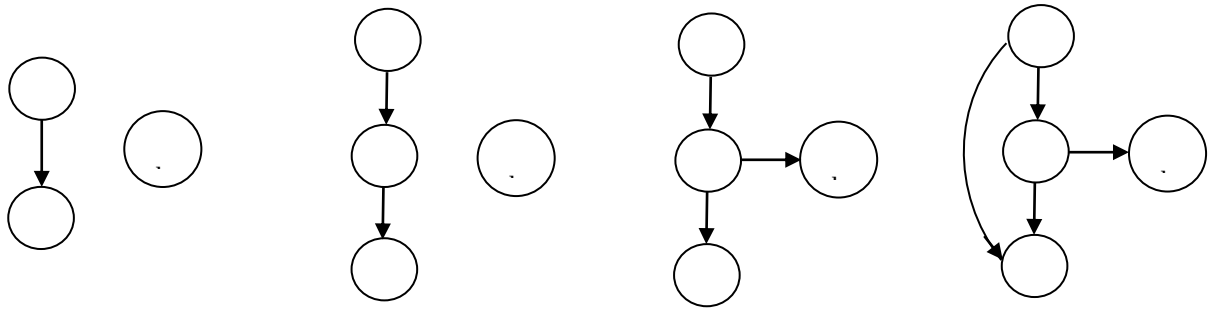
блоками ООС. Для сопряжения «l» с блоками ООС будем использовать связь – коннектор от исходного блока ООС к «l» и от «l» к блоку ООС, в который входит семантическая / пространственная связь (рис. 3б). При удалении «l» построим связь «l» с «tb» (рис. 3в) и добавим еще одну связь между «a» и «b», формализующую удаление прямой связи «l» между блоками «a» и «b» (рис. 3г).



а) **б)** **в)**

Рис. 2. Удаление блока ООС

Fig. 2. Deletion of OOS block



а) **б)** **в)** **г)**

Рис. 3. Удаление связи между блоками ООС

Fig. 3. Deletion of connectivity among OOS blocks

Очевидно, что построенные конструкции могут быть реализованы с применением аппарата онтологий, а сами операции удаления блоков ООС и связей между ними могут быть формализованы с дизъюнктом Хорна. Структура ООС (n-блочные ООС) представляет собой комбинацию попарно сопряженных блоков ООС, при этом полнота системы блоков и типов связей определена UML. Известно, что структура ООС взаимно-однозначно приводится к онтологической форме, при этом было показано, что операция удаления блоков и связей между ними может быть выражена дизъюнктом Хорна, и процесс низкоуровневой структурной модификации ООС также может быть выражен дизъюнктом Хорна.

■

Экспериментальная часть

На основе системы генерации кода многокомпонентных объектно-ориентированных структур [8] были разработаны:

- генераторы кода UI-компонентов;
- правила пространственного сопряжения UI-компонентов, обеспечивающие низкоуровневую структурную модифицируемость визуальной части UI-системы;
- правила семантического сопряжения UI-компонентов, обеспечивающие низкоуровневую структурную модифицируемость функциональной части UI-системы.

Фрагмент описания основных UI-компонентов и правил их пространственного сопряжения представлен в табл. 1. Правила семантического сопряжения позволяют выполнять сопряжение компонентов визуальной части UI-системы с компонентами, реализующими обработку системных и пользовательских событий, механизмы взаимодействия с СУБД, обработчиками DOM и т.д.

Разработана акторно-ориентированная система с CLI, которая:

- предоставляет доступ к онтологии UI-компонентов;
- осуществляет запуск и остановку процесса сопряжения генераторов кода;
- позволяет случайным образом позиционировать одни компоненты в области других;
- позволяет позиционировать одни компоненты в области других на основе параметрической настройки межкомпонентных расстояний;
- позволяет блокировать слои UI-компонентов во время модификации;
- поддерживает язык SparQL-запросов, с помощью которого возвращается конфигурация UI-системы.

Таблица 1.

Компоненты пользовательского интерфейса

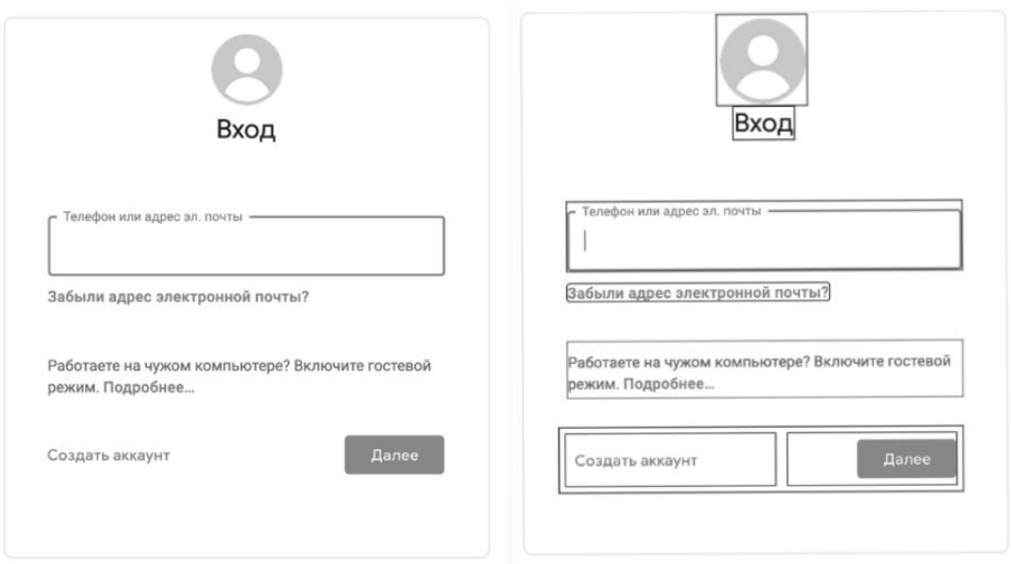
Table 1.

User Interface Components

Название компонента	**Композиция с другими компонентами**	**Возможность встраивания внутрь компонента**	**Визуальное представление компонента**
Компонент «Текст» (Label)	Сверху, снизу, справа, слева, «обертка»	Только «События» и «Стиль»	
Компонент «Блок» (div)	Сверху, снизу, справа, слева, «обертка», «вставка»	Любой компонент	
Компонент «Картинка» (Image)	Сверху, снизу, справа, слева, «обертка»	Только «События» и «Стиль»	
Компонент «Флажок» (Checkbox)	Сверху, снизу, справа, слева, «обертка»,	Только «События» и «Стиль»	
Компонент «Поле ввода» (Input)	Сверху, снизу, справа, слева, «обертка»,	Только «События» и «Стиль»	

Проведены эксперименты по синтезу множества конфигураций визуальной части UI-системы. На рис. 4а представлена одна из UI-систем, полученных в результате пространственной композиции UI-компонентов, часть которых, в свою очередь, также являются композитами.

Примером композита служит «Кнопка» (Button), являющаяся интерактивным компонентом, отображающим текст и изображения. Связь визуальной части с функциональными элементами в рамках настоящей работы не рассматривается. Синтезированная UI-система является многослойной, на рис. 4б показаны блоки (div), рендеринг которых выполнен цветом фона общего блока (UI-формы), представляющего собой контейнер, внутри которого размещены остальные элементы. Были выбраны пять конфигураций UI-систем, к которым применялась операция низкоуровневой структурной модификации, при этом время получения модификаций было ограничено (табл. 2).



а) **б)**

Рис. 4. Конфигурация UI-системы доступа к ресурсу

Fig. 4. Resource access UI system configuration

Таблица 2.

Результаты эксперимента по модификации UI-систем

Table 2.

Results of the experiment on modification of UI systems

Номер конфигурации	Количество полученных модификаций	Количество корректных модификаций	Количество модификаций с ошибками позиционирования	Количество модификаций с потерянными UI-компонентами
1	36	34 (94,4 %)	2 (5,6 %)	0 (0 %)
2	31	27 (87 %)	3 (9,7 %)	1 (3,3 %)
3	39	35 (89,7 %)	4 (10,3 %)	0 (0 %)
4	32	32 (100 %)	0 (0 %)	0 (0 %)
5	38	37 (97,3 %)	1 (2,7 %)	0 (0 %)

Результаты и перспективы

Рассмотренный процесс генерации UI–интерфейса отражает практическую возможность модификации объектно-ориентированных систем посредством представления системы в виде набора связанных компонентов. Показаны базовые компоненты, на основе которых могут быть построены другие более сложные компоненты. Подготовлены правила композиции таких элементов.

В результате синтеза и модификации UI-систем (табл. 2) были получены корректные UI-системы; в среднем в 93,68 %, в 0,66 % наблюдалась критическая потеря UI-компонента, а в 5,66 % имели место ошибки позиционирования, которые могли потенциально привести к отказам подсистем человеко-машинного взаимодействия.

Заключение

Рассмотрен процесс модификации объектно-ориентированных систем (ООС) в контексте структурных изменений. Данные изменения не нарушают базовые (низкоуровневые)

правила построения ООС. Представлен вариант модификации ООС с помощью внешней управляющей системы. Показано, что модификация любой ООС может быть представлена с помощью дизъюнкта Хорна. Проведены эксперименты по низкоуровневой структурной модификации и синтезу UI-систем.

Библиографический список

1. ГОСТ Р ИСО/МЭК 25010-2015, Национальный стандарт Российской Федерации. Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов [Электронный ресурс] // Режим доступа: https://docs.cntd.ru/document/1200121069 (Дата обращения 23.02.2022).
2. **Аристов, А.В.** Обеспечение интероперабельности систем формирования стандартизированных профилей [Текст] // Вестник Воронежского государственного технического университета. 2015. № 4. Т. 11. С. 40-43.
3. **Gamma, E.** Design Patterns: Elements of Reusable Object-Oriented Software [Text] / E. Gamma, R. Helm, R. Johnson, J. Vlissides. – Addison-Wesley Pub Co, 1995. – 395p.
4. **Жевнерчук, Д.В.** Алгебраические аспекты и структурно-параметрический синтез открытых информационных систем [Текст] / Д.В. Жевнерчук, Л.С. Ломакина // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT’18». 2018. Т. 2. С. 133-141.
5. **Жевнерчук, Д.В.** Обобщенный метод синтеза многокомпонентных интероперабельных структур на основе онтологии и недетерминированного конечного автомата [Текст] // Информационные технологии. 2019. № 2. С.67-74.
6. **Жевнерчук, Д.В.** Применение методов теории самоорганизации в задачах управления профилированием и конфигурированием вычислительных систем [Текст] / В.В. Кондратьев, Д.В. Жевнерчук // Доклады академии наук . 2014. Т. 459. № 4. С. 409-412.
7. **Жевнерчук, Д.В.** Принципы технической самоорганизации и структурно-параметрический синтез открытых информационных систем [Текст] // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS&IT’17». Т. 2. С. 167-175
8. **Жевнерчук, Д.В.** Семантическое моделирование генераторов программного кода распределенных автоматизированных систем [Текст] / Д.В. Жевнерчук, А.С. Захаров // Труды НГТУ им. Р.Е. Алексеева. 2018. № 1. С. 23-31.

Дата поступления
в редакцию: 21.03.2022