УДК 004.514
DOI: 10.26102/2310-6018/2025.50.3.041

Структурное моделирование графических пользовательских интерфейсов на основе алгебро-логических методов

В.О. Сапожников, А.В. Тарасов, Е.В. Кузнецова, А.С. Суркова, Д.В. Жевнерчук✉

Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация

Резюме. Работа посвящена актуальным вопросам синтеза средств человеко-машинного взаимодействия, в рамках которых рассматривается модель сопряжения компонентов графических пользовательских интерфейсов (ГПИ) на основе алгебро-логических методов. Компоненты ГПИ представляются в виде компонентов открытых информационных систем, имеющих стандартизированные интерфейсы, определяющие их пространственную совместимость. Для формализации компонентов ГПИ предлагается использовать семантические сети, при этом совместимость компонентов определяется правилами логического вывода, представленных в форме дизъюнкта Хорна. Приведено представление интегрированного визуального компонента «Именованное поле ввода» в виде семантической сети, содержащей описание пространственной совместимости входящих в его состав неделимых компонентов. Разработано расширение спецификации OpenAPI для решения проблемы унификации и стандартизации описания компонентов ГПИ, обеспечения интероперабельности инструментальных средств синтеза экранных форм и поддержки UX-тестирования. В статье представлены результаты синтеза цепочек геометрических фигур, имитирующих компоненты ГПИ, которые также могут быть представлены декларативно в виде семантических сетей, а, следовательно, и в формате RDF. Кроме самих компонентов семантические сети включают описание фильтров, с помощью которых можно управлять выбором способов пространственного сопряжения компонентов ГПИ.

Ключевые слова: человеко-машинное взаимодействие, графический пользовательский интерфейс, спецификация, компонент, дизъюнкт Хорна.

Для цитирования: Сапожников В.О., Тарасов А.В., Кузнецова Е.В., Суркова А.С., Жевнерчук Д.В. Структурное моделирование графических пользовательских интерфейсов на основе алгебро-логических методов. *Моделирование, оптимизация и информационные технологии.* 2025;13(3). URL: https://moitvivt.ru/ru/journal/pdf?id=2013 DOI: 10.26102/2310-6018/2025.50.3.041

Structural modeling of graphical user interfaces based on algebraic logic methods

O.V. Sapozhnikov, A.V. Tarasov, E.V. Kuznetsova, A.S. Surkova, D.V. Zhevnerchuk✉

Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation

Abstract. The work is devoted to topical issues of the synthesis of human-machine interaction tools, within the framework of which a model for interfacing components of graphical user interfaces (GUI) based on algebraic logic methods is considered. The components of GUI are presented as components of open information systems with standardized interfaces that determine their spatial compatibility. To formalize the components of the GUI, it is proposed to use semantic networks, while the compatibility of the components is determined by the rules of logical inference, presented in the form of a Horn disjunction. The description of the integrated visual component "Named input field" is presented in the form of a semantic network containing a description of the spatial compatibility of its constituent indivisible components. An extension of the OpenAPI specification has been developed to solve the

problem of unifying and standardizing the description of GUI components and ensuring the interoperability of tools for synthesizing screen forms and supporting UX testing. The article presents the results of the synthesis of chains of geometric shapes that mimic the components of GUI, which can also be presented declaratively in the form of semantic networks, and, consequently, in the RDF format. In addition to the components themselves, semantic networks include a description of filters that can be used to control the choice of ways to spatially interface GUI components.

Keywords: human-machine interaction, graphical user interface, specification, component, Horn's disjunction.

For citation: Sapozhnikov V.O., Tarasov A.V., Kuznetsova E.V., Surkova A.S., Zhevnerchuk D.V. Structural modeling of graphical user interfaces based on algebraic logic methods. *Modeling, Optimization and Information Technology*. 2025;13(3). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=2013 DOI: 10.26102/2310-6018/2025.50.3.041

Введение

Построение архитектуры информационных систем (ИС) осуществляется на основании сопоставления функциональных требований к информационной системе и их компонентам с функциями, предоставляемым типовыми классами программных продуктов и платформ. Решения, отраженные в архитектуре информационной системы, должны обеспечивать следующие свойства открытости ИС: возможность представления ее в виде совокупности автономных компонентов, взаимодействующих через стандартизированные интерфейсы, кроссплатформенность, масштабируемость, интероперабельность, отказоустойчивость, что улучшает её способность к интеграции с другими ИС, повышает степень адаптивности к изменениям внешней информационной среды и особенностям процессов человеко-машинного взаимодействия.

Одной из подсистем, реализующей процессы человеко-машинного взаимодействия, является графический пользовательский интерфейс (ГПИ) или Graphical user interface (GUI). Известны работы [1], в которых ГПИ представляет собой блочно-иерархическую систему, включающую 5 уровней, которые в свою очередь, представимы слоями, каждый из которых содержит блоки ввода данных от пользователя и блоки вывода информации.

Как правило, проектирование ГПИ – циклический процесс, включающий этапы синтеза версии ГПИ, выполнения UX-тестов, обработку метрик [2, 3], получаемых в результате прохождения UX-тестов, подготовку новой версии интерфейса, если не были достигнуты целевые значения метрик. На результаты синтеза графических пользовательских интерфейсов и выполнения UX-тестов влияют человеческий фактор, низкая квалификация исполнителей, попытки умышленного искажения значений UX-метрик.

Таким образом, тема настоящего исследования связана с формализацией ГПИ как открытой информационной системы и направлена на повышение степени автоматизации проектирования графических пользовательских интерфейсов, чем обусловлена ее актуальность. В ходе работы были решены следующие задачи:

– выполнена адаптация спецификации Open API 3.0, позволяющая представить ГПИ в качестве многокомпонентной структуры, элементы которой обладают стандартизированными интерфейсами;

– построены Open API спецификации основных элементов ГПИ;

– выполнена адаптация каркаса онтологии открытой информационной системы для описания ГПИ;

– предложена система правил логического вывода для автоматического определения возможности сопряжения графических элементов ГПИ;

– на основе предложенной модели построен генератор графических пользовательских интерфейсов;

– проведены экспериментальные исследования процесса синтеза графических пользовательских интерфейсов на основе их декларативного описания в виде семантической сети и с применением Open API 3.0 спецификации.

Материалы и методы

Известно, что блочно-иерархические системы могут быть представлены многокомпонентными интероперабельными структурами [4], формализуемыми с применением алгебро-логических методов [5]. При описании доменных ограничений, накладываемых на интерфейсы компонентов, могут применяться алгебраические типы данных[1]. Одним из известных широко применяемых на практике архитектурных стилей построения многокомпонентных систем является микросервисная архитектура [6, 7], в рамках которой для описания интерфейсов компонентов применяется спецификация OpenAPI 3.0[2].

В ходе анализа было выявлено, что проблема стандартизации описания элементов для открытых информационных систем (ОИС) является актуальной [8]. В большинстве случаев, в зависимости от используемого программного обеспечения и класса решаемых задач, разрабатывается собственная узконаправленная спецификация интерфейса прикладного программирования (Application Programming Interface, API).

В данной работе предлагается адаптация спецификации OpenAPI 3.0 для описания элементов ГПИ, являющегося подсистемой ОИС. Спецификации OpenAPI была выбрана вследствие ее распространенности, легкости машинной обработки, удобства человеческого восприятия, расширяемости. OpenAPI-спецификация определяет стандартизированное описание для REST API-сервисов, независимо от технологий при помощи которых они реализованы. Документация, составленная согласно спецификации OpenAPI, представляет собой JSON объект, который может быть представлен в формате JSON [9] или YAML документа[3]. Поскольку формат JSON-объекта представляет собой неупорядоченное множество пар «ключ: значение», то он является удобным для машинной обработки и приемлемым для человеческого восприятия.

Для спецификации OpenAPI характерно следующее:

– избыточность выразительных средств, ориентация на описание web-сервисов;

– однозначное соответствие между входными и выходными данными – request-response, что ограничивает ее применение для описания предметных областей.

В связи с этим OpenAPI не может быть напрямую использована для описания компонентов «черных ящик» с входными и выходными параметрами, а также для описания интерфейсной части библиотеки элементов GUI в форме «блок-свойство-домен».

[1] Jones S.P., Washburn G., Weirich S. Wobbly Types: Type Inference for Generalised Algebraic Data Types. Comp Geo at Tufts. URL: https://www.cs.tufts.edu/~nr/cs257/archive/simon-peyton-jones/MS-CIS-05-26.pdf (дата обращения: 16.06.2025).

[2] Open API Specification. Open API Initiative. URL: https://spec.openapis.org/oas/v3.0.0.html#openapi-specification (дата обращения: 16.06.2025).

[3] YAML Language Development Team. YAML Ain't Markup Language (YAML™) version 1.2. The Official YAML Web Site. URL: https://yaml.org/spec/1.2.2/ (дата обращения: 22.05.2025).

Результаты и обсуждение

Графический пользовательский интерфейс, как блочно-иерархическая структура, может быть представлен комбинацией графических элементов, положение которых определяется с помощью операций пространственного сопряжения [10], которые представлены в Таблице 1. Основные компоненты [10], на основе которых становится возможным создание ГПИ любой сложности путем их комбинирования, представлены в Таблице 2. Стоит отметить, что типы сопряжения «снизу» и «сверху» отражают варианты размещения компонентов на плоскости, а типы сопряжения «под» и «над» отражают варианты размещения компонентов, принадлежащих разным плоскостям (слоям GUI). В отличие от типа сопряжения «внутрь», типы сопряжения «под» и «над» могут привести к состоянию, в котором один компонент полностью или частично перекрывает другой. Тип сопряжения «снаружи» избыточен и может быть представлен посредством типа сопряжения «внутрь».

Таблица 1 – Операции пространственного сопряжения элементов
Table 1 – Operations of spatial pairing of elements

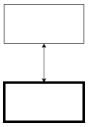
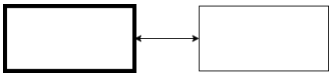
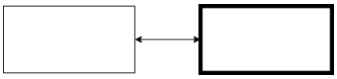
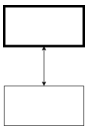
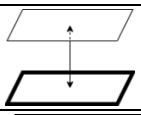
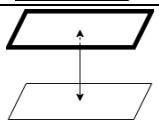
Название операции	**Схема**	**Описание**
Сопряжение **сверху**		Элемент будет расположен сверху от текущего.
Сопряжение **справа**		Элемент будет расположен справа от текущего.
Сопряжение **слева**		Элемент будет расположен слева от текущего.
Сопряжение **снизу**		Элемент будет расположен снизу от текущего.
Сопряжение **внутрь**		Элемент будет расположен внутри текущего.
Сопряжение **над**		Элемент будет расположен над текущим.
Сопряжение **под**		Элемент будет расположен под текущим

Таблица 2 – Основные элементы графического интерфейса
Table 2 – The basic elements of the graphical interface

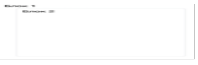
Название компонента	**Композиция с другими элементами**	**Возможность встраивания внутрь компонента**	**Визуальное представление компонента**
«Текст» (Label)	Сверху, снизу, справа, слева	Только «События» и «Стиль»	
«Блок» (Container)	Сверху, снизу, справа, слева, внутрь, над, под	Любой компонент	

Таблица 2 (продолжение)
Table 2 (continued)

«Картинка» (Image)	Сверху, снизу, справа, слева, внутрь	Только «События» и «Стиль»	
«Флажок» (Checkbox)	Сверху, снизу, справа, слева, внутрь	Только «События» и «Стиль»	
«Поле ввода» (Input)	Сверху, снизу, справа, слева, внутрь	Только «События» и «Стиль»	
«Событие» (Event)	Только внутрь	Нет	Нет
«Стили» (Style)	Только внутрь	Нет	Нет

Отдельно стоит отметить, что такие виды сопряжения как «над» и «под» доступны только компонентам, которые являются панелями (контейнерами) для остальных элементов. В Таблице 3 представлено описание полей разрабатываемой модификации OpenAPI для элементов ГПИ.

Таблица 3 – Поля OpenAPI для элементов ГПИ
Table 3 – OpenAPI fields for GUI elements

Имя поля	Тип	Описание
version	string	Семантический номер спецификации.
info	Info Object	Предоставление метаданных об элементе.
source	string	Источник/происхождение элемента.
properties	Properties Object	Информация о входных и выходных параметрах.
components	Component Object	Элемент для хранения различных схем для спецификации. Здесь же указываются пользовательские типы данных.
name	string	Название элемента.
version	string	Версия документа.
inputs	Array<PropertyObject>	Массив входных свойств.
outputs	Array<PropertyObject>	Массив выходных свойств.
schemas	Map<String, Array<TypeObject>>	Описание агрегатных типов данных.

Из спецификации OpenAPI были заимствованы следующие механизмы: описание примитивных типов, структур данных, использования перечислений, механизмы расширения спецификации и определения агрегатных типов.

Стоит отметить, что общая информация о графическом элементе представлена семантическим номером спецификации (version), метаданными о графическом элементе (info), происхождением графического элемента (source), информацией о входных/выходных параметрах (properties), ссылки на вспомогательные схемы и пользовательские типы данных (UDT). Объект свойств представлен массивами входных (inputs) и выходных (outputs) свойств, а описание агрегатных типов данных – тэгом schemas.

Для описания ГПИ воспользуемся каркасами онтологий ОИС [4], фрагмент которого представлен на Рисунке 1.

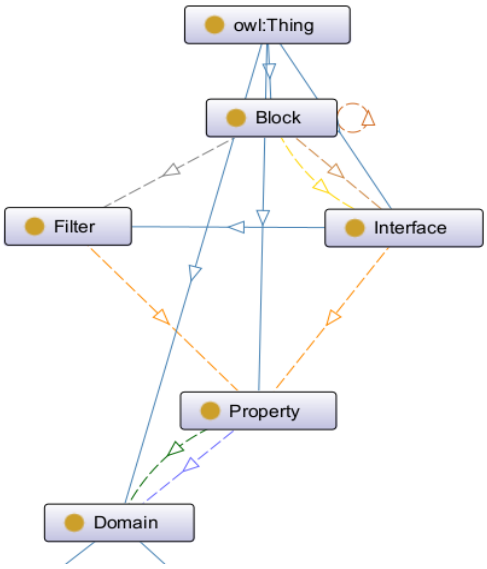


Рисунок 1 – Фрагмент онтологического каркаса ОИС
Figure 1 – Fragment of the ontological framework of the OIS

В таблице 4 представлено описание отношений (объектных свойств) между концептами owl: Thing, Block, Filter, Interface, Property, Domain.

Таблица 4 – Отношения между концептами фрагмента каркаса онтологии ОИС
Table 4 – Relations between the concepts of a fragment of the framework of the OIS ontology

Объект	Субъект	Объектное свойство	Комментарий
Block	Block	hasCompatible	Блоки совместимы и могут располагаться друг с другом.
Block	Interface	hasInputInterface	Блок имеет входной интерфейс.
		hasOutputInterface	Блок имеет выходной интерфейс.
Block	Filter	hasFilter	Блок имеет фильтр.
Interface	Property	hasProperty	Интерфейс имеет свойство.
Filter	Property	hasProperty	Фильтр имеет свойство (будет воздействовать на свойство).
Property	Domain	hasDomain	Связывает свойство интерфейса со значением из доменного ограничения.
		hasPermissible Domain	Связывает свойство фильтра и область допустимых значений.

Для определения совместимости компонентов онтология содержит правила логического вывода. Рассмотрим два ключевых правила в форме дизъюнкта Хорна [11]: правило сопряжения компонентов без фильтрации:

$$(?\,b1\ hasOutputInterface\ ?\,i1) \cap (?\,b2\ hasInputInterface\ ?\,i2) \cap \quad (1)$$
$$noValue(?\,b2\ hasFilter) \cap notEqual(?\,b1, ?\,b2) \rightarrow (?\,b1\ hasCompatible\ ?\,b2)$$

и правило сопряжения компонентов с фильтрацией:

$$(?\,b1\ hasOutputInterface\ ?\,i1) \cap (?\,i1\ hasProperty\ ?\,p1) \cap$$
$$(?\,p1\ hasDomain\ ?\,d1) \cap (?\,b2\ hasInputInterface\ ?\,i2) \cap (?\,b2\ hasFilter\ ?\,fltr) \cap \quad (2)$$

$$(?\,fltr\ hasProperty\ ?\,fltrProp) \cap (?\,fltrProp\ hasPermissibleDomain\ ?\,prmsD) \cap$$

$$listContains(?\,prmssD, ?\,d1) \cap notEqual(?\,b1, b2) \rightarrow (?\,b1\ hasCompatible\ ?\,b2)$$

Правило (1) интерпретируется следующим образом: если блок *b1* имеет выходной интерфейс *i1* и блок *b2* имеет входной интерфейс *i2* и блок *b2* не имеет свойства «имеет фильтр» и блок *b1* не является блоком *b2,* тогда блок *b1* совместим с блоком *b2.*

Правило (2) интерпретируется следующим образом: если блок *b1* имеет выходной интерфейс *i1* и выходной интерфейс *i1* ассоциирован со свойством *p1* и свойство *p1* имеет доменное ограничение *d1* и блок *b2* имеет входной интерфейс *i2* и блок *b2* имеет фильтрующий интерфейс *fltr* и фильтрующий интерфейс *fltr* ассоциирован со свойством $fltrProp$ и свойство $fltrProp$ имеет допустимый домен $prmsD$ и домен $prmsD$ является частью домена *d1* и блок *b1* не равен блоку *b2*, тогда блок *b1* совместим с блоком *b2*.

Рассмотрим в качестве примера описание элемента «Label» согласно предложенной спецификации (Рисунок 2).

```
{
        “version”: “1.0”,
        “info”: { “name”: “Label”, “version”: “1.0”},
        “sources”: “HTML”,
        “properties”: {
                inputs: [
                        { “type”: “string”},
                        { “type”: “string”, “format”: “style”},
                        { “type”: “string”, “format”: “spatialConnections”,
                                “enum”: [ “topOf”, “rightOf”, “bottomOf”, “leftOf”, “insert”]
                        }
                ],
                outputs: [
                        { “type”: “string” },
                        {  “type”: “string”, “format”: “spatialConnections”,
                                “enum”: [ “topOf”, “rightOf”,  “bottomOf”, “leftOf”]
                        }
                ]
        }
}
```

Рисунок 2 – Описание элемента «Label»
Figure 2 – Description of the "Label" element

На Рисунке 3 представлена многокомпонентная структура «Именованное поле ввода».

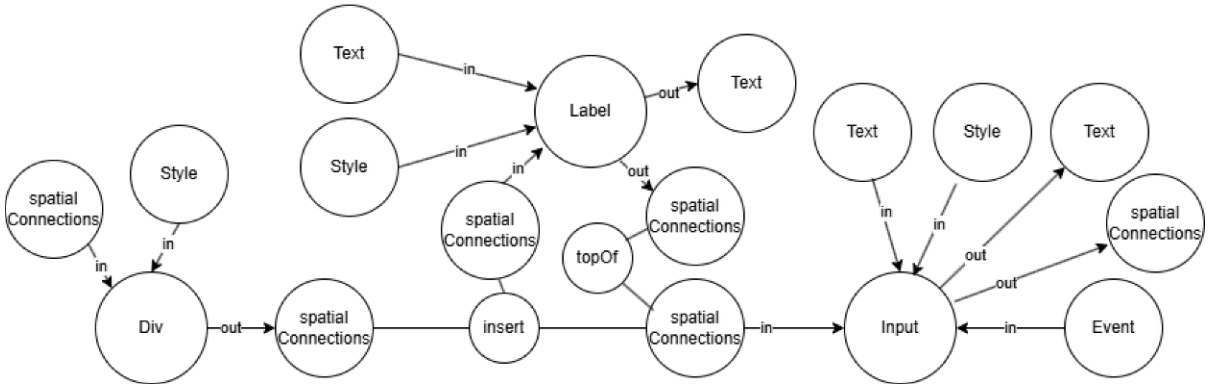


Рисунок 3 – Многокомпонентная структура «Именованное поле ввода»
Figure 3 – Multicomponent structure «Named Input Field»

Отметим, что в качестве средств описания компонента используется html, чем обусловлена замена общего компонента «Container» на специфический «Div», являющийся контейнером блочного типа, который определяет раздел или секцию в html документе.

Сопряжение «insert», являющееся выходным для компонента «Container» и входным для компонентов «Label» и «Input», обозначает, что последние элементы встраиваются в «Container». Сопряжение «topOf», которое является выходным для элемента «Label» и входным для элемента «Input», обозначает, что «Label» находится над «Input».

Стоит отметить, что поскольку основные элементы (Таблица 2) могут быть описаны по предложенной спецификации (Таблица 3), то многокомпонентная структура «Именованное поле ввода» автоматически может быть выражена по той же спецификации.

Рассмотрим процесс синтеза ГПИ, представляющего один слой, с расположенными на нем фигурами: треугольник, квадрат, круг, имитирующие элементы ГПИ. При этом каждая из фигур может иметь свой уникальный цвет. На Рисунке 4 представлен пример ГПИ.

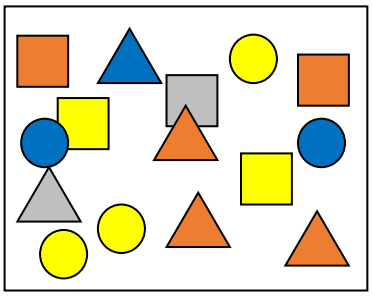


Рисунок 4 – Пример экранной формы тестового ГПИ
Figure 4 – An example of an on-screen form for a test GUI

Рассмотрим описание графических элементов ГПИ (Рисунок 5). Домены «Цвет» («Color») и «Тип Фигуры» («Figure») являются подклассами «Domain». Для класса «Color» определены три индивида, представляющие собой элементы доменного ограничения типа «enum»: зеленый, синий, красный. Для класса «ТипФигуры»: треугольник, квадрат и круг.

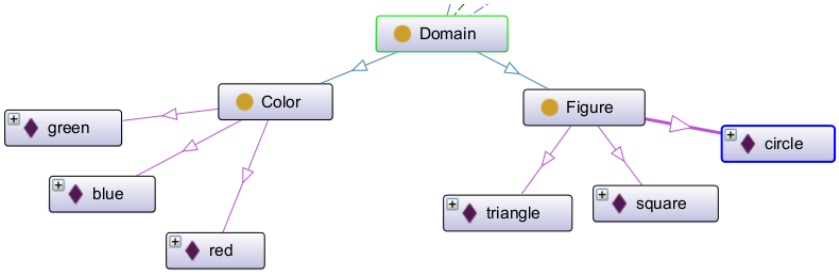


Рисунок 5 – Фрагмент описания тестовой ГПИ
Figure 5 – A fragment of the description of the test GUI

Каждый компонент имеет следующие интерфейсы:

– входной инициализирующий интерфейс, через который задаются свойства компонента (цвет и фигуры) при создании;

– входной интерфейс, который принимает данные совместимого компонента;

– выходной интерфейс, который передает данные текущего компонента, получаемые при его создании через входной инициализирующий интерфейс.

Помимо этого, компонент может иметь фильтр, который накладывает ограничения на область значений входного интерфейса – ограничение на возможные совместимые компоненты.

Проверка того, что элементы не равны друг другу, нужна для запрета «петель», когда элемент совместим сам с собой, поскольку это не предусмотрено условиями данной задачи.

В качестве простого примера рассмотрим процесс сборки графических пользовательских интерфейсов, включающий базовый слой-контейнер и три элемента пользовательских интерфейсов, представляющих собой фигуры разных типов и разных цветов: красный квадрат, зеленый треугольник, синий круг.

Компонент, моделирующий элемент красный квадрат не имеет фильтра, компонент инициализируется свойствами, которые имеют значения red и square. Эти же свойства являются выходными для компонента.

Компонент, описывающий зеленый треугольник также не имеет фильтра. Данный компонент инициализируется свойствами, которые имеют значения green и triangle. Эти же свойства являются выходными для компонента.

Компонент, представляющий синий круг, обладает фильтрующим интерфейсом, который пропускает два значения свойства Color: red, blue и два значения свойства Figure: square, circle. Компонент инициализируется значением blue для свойства Color и circle для свойства Figure. Эти же свойства являются выходными для компонента.

Процесс синтеза пользовательского интерфейса сводится к построению многокомпонентных структур, образуемых компонентами с помощью операции сопряжения через интерфейсы типа «out» и «in». Фильтрующие интерфейсы, как уже отмечалось, не участвуют в операции сопряжения напрямую, но они позволяют ограничить домен выходных и входных интерфейсов подмножеством значений. В рамках экспериментальной проверки многокомпонентная структура может интерпретироваться следующим образом: если два компонента в цепочке компонентов совместимы, то компонент, формирующий поток данных визуально располагается левее компонента, принимающего поток данных (Рисунок 6).

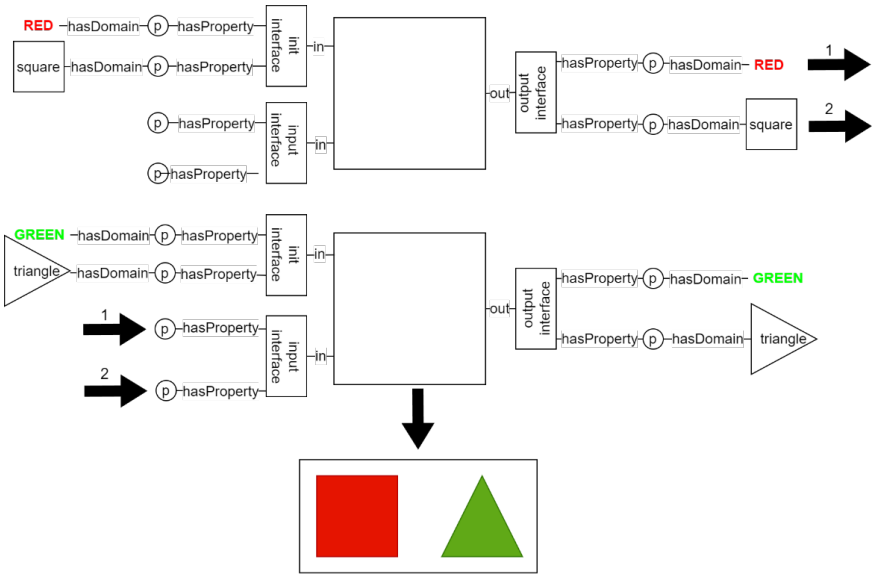
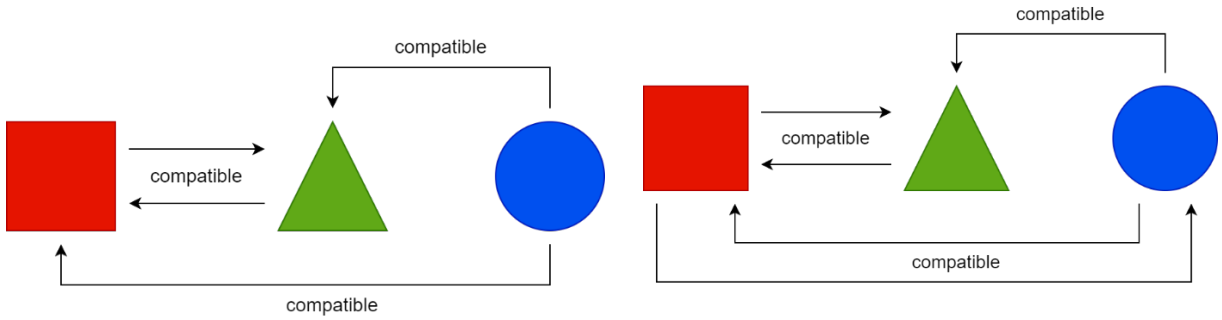


Рисунок 6 – Соответствие пары совместимых компонентов расположению элементов ГПИ
Figure 6 – Matching a pair of compatible components to the location of GUI elements

Рассмотрим схему совместимости элементов GUI (Рисунок 7*а*), которая возникает при применении правила сопряжения компонентов без фильтрации (1). Красный квадрат и зеленый треугольник являются взаимно совместимыми, то есть могут быть расположены на базовом слое и справа и слева друг от друга, так как не имеют фильтров. Синий круг может быть расположен только слева, поскольку также совместим с остальными фигурами, но при этом ни одна фигура не совместима с синим кругом, потому что у него есть фильтрующий интерфейс, который по умолчанию ограничивает домен входных свойств пустым множеством.

После применения правила для элементов с фильтрами (2) с синим кругом становится совместим красный квадрат, так как его выходные данные входят в область допустимых значений, ограниченных фильтром. Красный квадрат и синий круг становятся взаимно совместимыми (Рисунок 7*б*).



а) *б*)

Рисунок 7 – Схема совместимости элементов после применения правил
Figure 7 – The scheme of compatibility of elements after application of rules

При применении правила (1) синтез ГПИ выполняется в рамках следующих ограничений:

– фильтрующие интерфейсы не учитываются;

– все блоки без фильтров (красный квадрат и зеленый треугольник) взаимно совместимы и могут располагаться слева и справа друг от друга;

– синий круг, хотя и совместим по общим условиям с другими фигурами, из-за наличия фильтра с пустым доменом входных значений не получает обратно совместимых связей.

При применении правила (2) синтез ГПИ выполняется в рамках следующих ограничений:

– учитываются фильтрующие интерфейсы и доменные ограничения;

– фильтр синего круга пропускает значения цвета {red, blue} и фигуры {square, circle};

– красный квадрат соответствует этим ограничениям, поэтому связь между красным квадратом и синим кругом становится двунаправленной: оба блока взаимно совместимы.

Получаемый в результате ГПИ формируется на основе онтологического представления, тогда он представим в формате RDF[4].

[4] Curé O., Blin G. *RDF Database Systems: Triples Storage and SPARQL Query Processing.* Waltham: Morgan Kaufmann; 2014. 256 p.

Заключение

В работе проведен структурный анализ графических пользовательских интерфейсов, как класса ОИС.

Предложена модель ГПИ, представляющая собой многокомпонентные структуры, в которых каждый компонент формализует слой или управляющий элемент GUI. Особенностью модели является существование входных/выходных интерфейсов у элементов и слоев, которые обеспечивают потоки данных между компонентами. Существование потока данных дает информацию о пространственном расположении компонентов друг относительно друга. Входные/выходные интерфейсы относятся к фильтруемым, так как допускают динамическое ограничение области допустимых значений. Множество принимаемых значений можно ограничить посредством фильтрующих интерфейсов. Построение интегрированных интерфейсов основывается на алгебраических типах данных.

Предложена адаптация спецификации OpenAPI 3.0 для описания элементов ГПИ, а также правила логического вывода и операций пространственного сопряжения.

Полученные результаты могут быть использованы при построении средств интеллектуальной поддержки процесса проектирования графических пользовательских интерфейсов, а также библиотек и плагинов, применяемых в редакторах GUI.

В дальнейшем планируется адаптация спецификации Open API 3.0 для формализации элементов такого средства человеко-машинного взаимодействия как аудио-интерфейс, а также расширение списка графических элементов для ГПИ, применяемых в АСУ ТП, панелях управления транспортными средствами и оборудованием промышленных предприятий.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Garrett J.J. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Berkeley: New Riders; 2011. 172 p.
2. Finstad K. The Usability Metric for User Experience. *Interacting with Computers.* 2010;22(5):323–327. https://doi.org/10.1016/j.intcom.2010.04.004
3. Синица С.А. Оценка качества взаимодействия пользователя с ИИ-интерфейсами: когнитивные нагрузки, UX-метрики и пользовательская лояльность. *Международный журнал гуманитарных и естественных наук.* 2025;(6–2):18–22. https://doi.org/10.24412/2500-1000-2025-6-2-18-22
   Sinitsa S.A. Evaluating the Quality of User Interaction with AI Interfaces: Cognitive Loads, UX Metrics, and User Loyalty. *International Journal of Humanities and Natural Sciences.* 2025;(6–2):18–22. (In Russ.). https://doi.org/10.24412/2500-1000-2025-6-2-18-22
4. Жевнерчук Д.В. Обобщённый метод синтеза многокомпонентных интероперабельных структур на основе онтологии и недетерминированного конечного автомата. *Информационные технологии*. 2019;25(2):67–74. https://doi.org/10.17587/it.25.67-74
   Zhevnerchuk D.V. A Generalized Method for Synthesizing Multicomponent Interoperable Structures Based on Ontology and a Nondeterministic Finite State Machine. *Information Technologies.* 2019;25(2):67–74. (In Russ.). https://doi.org/10.17587/it.25.67-74
5. Титов А.В. Теория сложности в алгебро-логическом моделировании систем управления. В сборнике: *Управление развитием крупномасштабных систем – MLSD'20: труды Тринадцатой Международной научно-технической конференции, 28–30 сентября 2020 года, Москва, Россия.* Москва: Институт проблем управления

им. В.А. Трапезникова РАН; 2020. С. 229–239. https://doi.org/10.25728/mlsd.2020.0229
6. Busi S.P. Understanding Microservices Architecture: A Comprehensive Guide. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology.* 2025;11(1):1440–1447. https://doi.org/10.32628/CSEIT251112144
7. Bello-Trejo S., Limón X., Ocharán-Hernández J.O., Hernández-González L.A. System-Oriented Testing on the Microservices Architecture: A Systematic Literature Review. In: *2024 12th International Conference in Software Engineering Research and Innovation (CONISOF), 28 October – 01 November 2024, Puerto Escondido, Mexico.* IEEE; 2024. P. 127–136. https://doi.org/10.1109/CONISOFT63288.2024.00026
8. Гуляев Ю.В., Журавлев Е.Е., Олейников А.Я. Методология стандартизации для обеспечения интероперабельности информационных систем широкого класса. *Журнал радиоэлектроники*. 2012;(3). URL: http://jre.cplire.ru/win/mar12/2/text.pdf
9. Marrs T. *JSON at Work: Practical Data Integration for the Web.* Sebastopol: O'Reilly Media; 2017. 376 p.
10. Жевнерчук Д.В., Захаров А.С. Модель и лингвистическое обеспечение низкоуровневой структурной модификации объектно-ориентированных систем. *Труды НГТУ им. Р.Е. Алексеева*. 2022;(2):7–16. https://doi.org/10.46960/1816-210X_2022_2_7
Zhevnerchuk D.V., Zakharov A.S. Model and Linguistic Support of Low-Level Structural Modification of Object-Oriented Systems. *Trudy NGTU im. R.E. Alekseeva*. 2022;(2):7–16. (In Russ.). https://doi.org/10.46960/1816-210X_2022_2_7
11. Caferra R. *Logic for Computer Science and Artificial Intelligence*. London, Hoboken: ISTE, John Wiley & Sons; 2011. 523 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Сапожников Владислав Олегович, ассистент кафедры «Вычислительные системы и технологии», Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация.
e-mail: vlad.sapozhnickof@gmail.com
ORCID: 0009-0000-5696-9951

Vladislav O. Sapozhnikov, Assistant Professor at the Department of Computer Systems and Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation.

Тарасов Александр Васильевич, ассистент кафедры «Вычислительные системы и технологии», Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация.
e-mail: alexandervtarasovddd@gmail.com
ORCID: 0009-0009-6868-2937

Alexandr V. Tarasov, Assistant Professor at the Department of Computer Systems and Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation.

Кузнецова Евгения Владимировна, аспирант кафедры «Вычислительные системы и технологии», Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация.
e-mail: ewgesha2008@yandex.ru
ORCID: 0009-0004-5991-1412

Ewgenia V. Kuznetsova, Postgraduate at the Department of Computer Systems and Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation.

Суркова Анна Сергеевна, доктор технических наук, доцент, профессор кафедры «Вычислительные системы и технологии», Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация.
e-mail: ansurkova@yandex.ru
ORCID: 0000-0003-0018-9053

Anna S. Surkova, Doctor of Engineering Sciences, Docent, Associate Professor at the Department of Computer Systems and Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation.

Жевнерчук Дмитрий Валерьевич, доктор технических наук, доцент, профессор кафедры «Вычислительные системы и технологии», Нижегородский государственный технический университет им. Р.Е. Алексеева, Нижний Новгород, Российская Федерация.
e-mail: zhevnerchuk@yandex.ru
ORCID: 0000-0002-6306-0893

Dmitriy V. Zhevnerchuk, Doctor of Engineering Sciences, Docent, Associate Professor at the Department of Computer Systems and Technologies, Nizhny Novgorod State Technical University n.a. R.E. Alekseev, Nizhny Novgorod, the Russian Federation.

Статья поступила в редакцию 07.07.2025; одобрена после рецензирования 13.08.2025; принята к публикации 26.08.2025.

The article was submitted 07.07.2025; approved after reviewing 13.08.2025; accepted for publication 26.08.2025.