УДК 004.514

А.П. ГОРДИЕНКО, канд. техн. наук, доц., доц. каф. „Информационные системы”, Госуниверситет - УНПК, Орел, Россия

ПУТИ СОВЕРШЕНСТВОВАНИЯ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА ДЛЯ МАШИНОСТРОИТЕЛЬНЫХ САПР

В статье рассматриваются пути совершенствования графического пользовательского интерфейса для машиностроительных САПР: использование новых метафор диалога, применение средств поддержки ограничений целостности и алгоритмов интерпретации действий пользователя в зависимости от визуального контекста. Показана целесообразность использования для этого методов чистого функционального программирования.

У статті розглядаються шляхи вдосконалення графічного інтерфейсу користувача для машинобудівних САПР: використання нових метафор діалогу, застосування засобів підтримки обмежень цілісності і алгоритмів інтерпретації дій користувача в залежності від візуального контексту. Показана доцільність використання для цього методів чистого функціонального програмування.

The article discusses ways to improve the graphical user interface for CAD engineering: the use of new metaphors of dialogue, use of support constraints and algorithms to interpret user actions depending on the visual context. The feasibility of using of the pure functional programming is shown.

Введение. Успешность применения САПР в большой степени зависит от качества ее пользовательского интерфейса (ПИ) – подсистемы, отвечающей за ее взаимодействие с оператором-проектировщиком. Неудобный ПИ может сделать малопригодными даже самые хорошие прикладные программы. На наш взгляд, в коллектив разработчиков САПР требуется включать специалистов по ПИ, среди которых должны быть как минимум автор диалога и специалист по эргономике.

В данной статье предлагается несколько направлений совершенствования ПИ. Во-первых, предлагается начать разработку с проектирования метафоры диалога. Здесь особая роль отводится специалистам по интерактивным методам и эргономике. Во-вторых, ПИ должен обеспечивать взаимодействие не только с отдельными компонентами прикладной модели, но и учитывать их взаимные соотношения. С этой целью в ПИ должны быть введены средства поддержки целостности данных. В-третьих, действия оператора должны интерпретироваться системой в зависимости от визуального контекста – состояния диалога и прикладной модели, представленного пользователю в графическом виде. Таким образом, можно существенно сократить число управляющих элементов ПИ и сделать взаимодействие с САПР более очевидным и понятным.

Кроме этого, при разработке ПИ необходимо использовать формальные спецификации логики диалога и важнейших структур данных. Для этого целесообразно использовать методы чистого функционального программирования, позволяющие строить доказательства корректности алгоритмов.

Целью данной работы является повышение качества ПИ за счет использования новых метафор диалога, интерактивных методов поддержки целостности прикладных моделей и анализа контекста диалога в процессе проектирования.

Разработка новых метафор. Метафора ПИ по аналогии с метафорами естественного языка определяет такое представление прикладной модели и объектов ПИ, при котором сложные и абстрактные компоненты замещаются более простыми и наглядными образами, имеющими аналогию с предметами реального мира. Применение метафор исключает необходимость непосредственного однозначного отображения внутренней структуры системы, и дает упрощенное, более понятное и привычное для конечного пользователя представление тех аспектов прикладной модели, которые необходимы ему для работы.

Для ГПИ машиностроительных САПР больше всего подходит метафора модели мира, которая предполагает, что на экране изображен образный мир объектов, имеющий свои законы и поведение. Пользователь взаимодействует с этим миром простыми действиями, в основном указывая и перемещая объекты. Мир реагирует перемещением объектов, изменением их формы, размеров, удалением объектов или визуализацией новых.

Метафора модели мира имеет свои корни в принципах непосредственного манипулирования, разработанных Б. Шнейдерманом [1]:

1) все интересующие пользователя объекты постоянно представлены на экране;

2) пользователь не строит сложных синтаксических форм (последовательностей операций), а просто перемещает объекты или выбирает их мышью;

3) все действия пользователя обратимы, а реакция на них прикладных модулей – мгновенна.

Применение принципов прямого манипулирования произвело революцию в области ГПИ. Однако при реализации сложных операций в графическом редакторе приходится возвращаться к командным языкам. Например, для построения дуги окружности обычно используют команды, в которых заключается способ задания дуги и необходимые для него параметры. Поэтому принципы прямого манипулирования нуждаются в доработке: необходимо включить в ГПИ возможность манипулирования более абстрактными свойствами объектов, которые не сводятся к простым количественным изменениям значений атрибутов. Эти свойства касаются принадлежности элементов классам и различным соотношениям между объектами. Если добавить в ГПИ возможность пошагового изменения класса объекта по направлению род – вид и обратно, то получится интерфейс с операциями развития и деградации объектов. Будем говорить, что такой интерфейс использует метафору эволюционного развития. Ее смысл в том, что чертеж строится из объектов, являющихся экземплярами классов, организованных в родовидовую иерархию. Небольшой фрагмент такой иерархии показан на рис. 1. На ее вершине располагается класс – линия, определяющий графические объекты, имеющие начало и конец. Графически их можно представить отрезком прямой. Линия может стать дугой, если задать радиус, может – кривой, если задать управляющие точки. Такое преобразо-

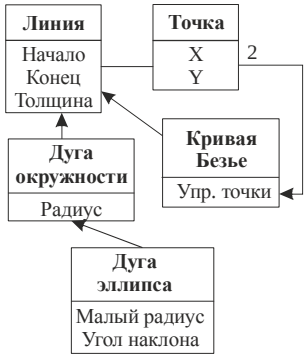

Рис. 1. Фрагмент иерархии классов графических объектов

вание можно задать непосредственным действием мышью. При работе с таким ПИ пользователь, имея минимальные предварительные знания, может экспериментальным путем изучить остальные возможности системы.

Будем считать, что одно действие пользователя имеет 4 фазы: перемещение, нажатие кнопки, перемещение с нажатой кнопкой, отпускание кнопки. Таким образом, таким действием можно ввести две точки или переместить один объект. Это значит, что объекты, которые для своего определения требуют не более двух точек, можно строить непосредственно. Из числа объектов, приведенных на рис. 1, так можно построить только линию. Как быть с более сложными объектами? – Можно применить метафору эволюции объекта. Например, отрезок прямой можно превратить (эволюционировать) в дугу окружности, если его дополнить новым атрибутом, например, третьей точкой или радиусом. Визуально это выглядит так: на первой фазе действия пользователя, когда он подводит мышь к какому-либо объекту, объект становится активным, как показано на рис. 2, а. Для него высвечиваются дополнительные построения (например, пиктограммы), обозначающие направления эволюции. Перемещение пиктограммы 4 вызывает эволюцию линии в дугу. Точка, в которой будет отпущена кнопка мыши, станет третьей точкой, определяющей дугу окружности. Аналогично из линии можно получить кривую Безье, переместив пиктограмму 3 и, таким образом, добавив управляющую точку. Воздействуя на пиктограммы 1 и 2 и перемещая отрезок, можно редактировать объект, не меняя его класса.

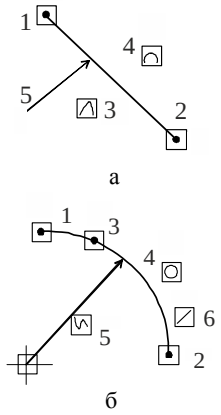

Рис. 2. Изображение выделенного объекта:
а – линии;
б – дуги окружности

На рис. 2, б показана активная дуга окружности. Ее можно превратить в кривую (пиктограмма 5) или в окружность (пиктограмма 4). Она деградирует в линию, если нажать пиктограмму 6. Остальные пиктограммы служат для простого изменения атрибутов дуги.

Важно отметить, что логику такого диалога можно определить строго, используя чистое функциональное программирование [2]. Но эта метафора все-таки имеет ограничение в случае, когда необходимо выполнить сложные структурные преобразования объектов или накладывать сложные ограничения целостности. Тогда применима метафора ассоциативного взаимодействия объектов. Ее суть в

том, что объекты ПИ наделяются способностью взаимодействовать друг с другом. Для этого с каждым объектом связывается шаблон возможных вариантов взаимодействия. Эта метафора описана в работах [3, 4]. При взаимодействии объектов происходит поиск возможных геометрических соотношений между ними. Если найденных вариантов окажется много, то можно использовать методы логического вывода, выбирающие наиболее подходящие соотношения.

Используя предлагаемые метафоры, можно сформулировать новый метод проектирования ПИ. Исходными данными для проектирования будут: концептуальная схема графических примитивов; набор команд, которые должна выполнять система. Алгоритм проектирования заключается в следующем. Из набора команд выделяем те, которые можно реализовать на основе принципа прямого манипулирования. Они могут быть реализованы в рамках парадигмы MVC. Из оставшейся части выделяем те команды, которые получаются из уже реализованных команд добавлением или перемещением одной точки в соответствии с метафорой эволюционного развития. Процесс заканчивается, если будут реализованы все команды. Если окажется, что какие-либо команды не могут быть реализованы на основе уже существующих, то возможны два варианта действий: если для выполнения команды нужно определить дополнительно более одной точки, то вводим в концептуальную схему промежуточный примитив; иначе определяем шаблон ассоциативного взаимодействия и связываем его с новой пиктограммой.

Использование метафор эволюционного развития и ассоциативного взаимодействия объектов позволило в рамках принципа прямого манипулирования строить сложные графические объекты с заданными геометрическими соотношениями, действующими на их компонентах. Отказ от использования командных языков позволил упростить реализацию пользовательского интерфейса.

Интерактивное задание геометрических соотношений. В большинстве случаев при подготовке графических документов требуется не только строить графические элементы, такие как отрезки прямой, дуги окружности и кривые, но и учитывать их геометрические соотношения. Может потребоваться, чтобы некоторые отрезки были параллельными, другие – перпендикулярными, и все они должны иметь определенные размеры. Эти соотношения накладывают ограничения на форму графических элементов. Таким образом, ПИ должен давать возможность оперировать такими ограничениями.

В настоящее время развивается целое направление в программировании – программирование на основе ограничений (constraints programming). Ограничения в графических системах впервые были применены более 30 лет назад. Одной из первых систем, использовавших язык с ограничениями и объектно-ориентированный подход в интерактивной графике, является Sketchpad [5], разработанный Сазерлендом. Вслед за ней появился целый ряд систем, таких как ThingLab [6], Juno [7], Amulet [8] и другие. Опыт создания и применения графических систем с ограничениями показал их эффективность и практическое значение. Примером современной системы, использующей ограничения, может служить чертежно-графическая система “Компас”.

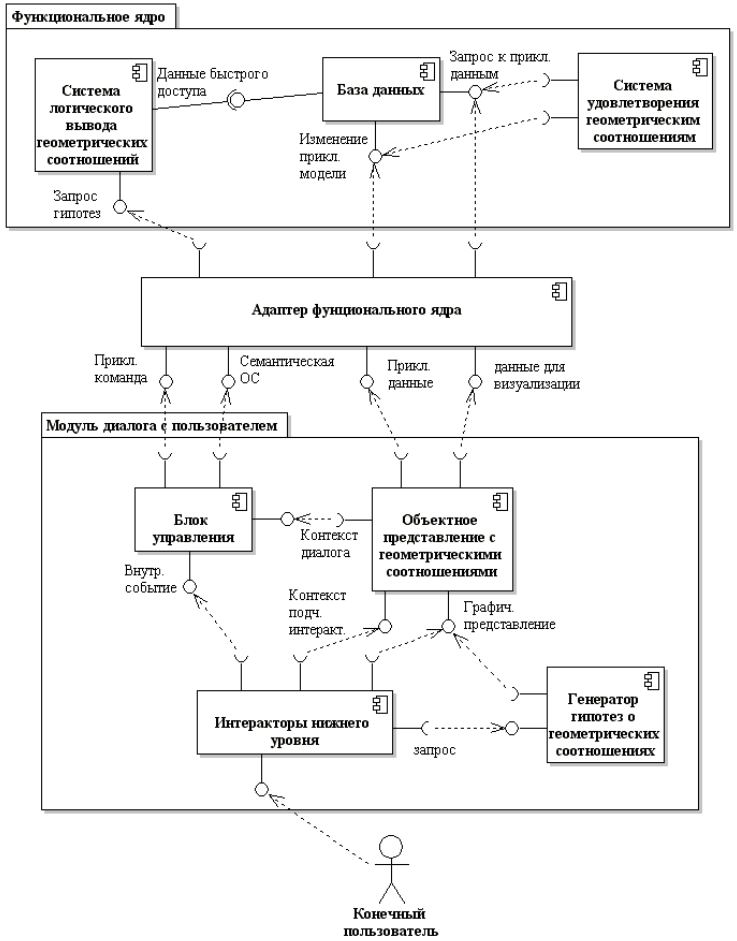

Рис. 3. Структура графического редактора, использующего геометрические соотношения

На рис. 3 средствами языка UML показана структура графического редактора с включенными в нее средствами поддержки геометрических соотношений. Эта структура основывается на эталонной модели интерактивной системы Arch, которая обсуждалась на международной конференции EWHCI'92, проходившей в России в Санкт-Петербурге в 1992 г. [9]. Анализ моделей графического пользовательского интерфейса проведен нами в работах [10, 11]. Модель Arch обеспечивает три независимых уровня представления данных:

1) прикладные данные, например, описание редактируемого чертежа;

2) данные, относящиеся к вводу/выводу: пиксели, курсоры, графические примитивы и т.п., называемые данными логических устройств ввода (ЛУВ) и

графической системы;

3) данные, относящиеся к визуальному представлению интерфейса.

В структуре графического редактора выделены три части: модуль диалога с пользователем, получающий входные события от пользователя и обеспечивающий визуальное представление прикладных данных и объектов ПИ; функциональное ядро - базу прикладных данных, хранящую проектируемую модель и адаптер функционального ядра, который является связующим звеном между диалоговой компонентой и функциональным ядром. Последний обеспечивает разделение прикладных модулей и ПИ и таким образом дает возможность независимой работы прикладных программистов и разработчиков ПИ.

Рассмотрим подробнее структуру модуля диалога с пользователем. Непосредственно ввод и вывод информации осуществляется интеракторами нижнего уровня. Идея интерактора сформировалась под влиянием объектно-ориентированной парадигмы программирования как обобщение логических устройств ввода – вывода, изначально определенных в стандарте GKS [12]. Интерактор осуществляет обмен информацией между пользователем и прикладной базой данных. Он имеет локальное состояние, способен участвовать в событиях окружения и, вследствие этого, менять свое состояние и визуальное представление. К настоящему времени уже разработан ряд моделей интеракторов, анализ которых дан нами в [13].

Объектное представление является структурированной совокупностью графических примитивов, отражающей графическое представление прикладной модели и объектов пользовательского интерфейса. На множестве графических примитивов заданы геометрические соотношения.

Блок управления получает поток внутренних событий от подчиненных интеракторов и формирует контекст диалога – информацию о текущем состоянии объектного представления. Помимо этого он во входной последовательности событий распознает прикладные команды и передает их адаптеру функционального ядра. Блок управления является отдельной задачей, выполняющейся параллельно с другими интеракторами и функциональным ядром. Синхронизация между ними осуществляется через общие события. Логика диалога определяется выражениями поведения, написанными на языке CSP [14] с использованием операторов, описанных в [2].

Генератор гипотез о геометрических соотношениях сопоставляет вводимые пользователем данные с объектным представлением и выявляет возможные ограничения, которые затем передаются функциональному ядру.

Функциональное ядро – база прикладных данных, хранящая проектируемую модель, в которой важное место занимают средства, обеспечивающие целостность данных. Ограничения целостности формируются системой логического вывода геометрических соотношений, которая отбирает из списка возможных ограничений те из них, которые непротиворечиво и достаточно доопределяют прикладную модель. Система удовлетворения геометрическим соотношениям (в случае внесения изменений в прикладную модель) приводит

в соответствие заданным ограничениям все остальные прикладные данные. Например, если пользователь ставит размер на чертеже, то добавляется новое ограничение – принадлежность прямой или окружности (в зависимости от типа размера). Если ей все ограничения удовлетворить не удастся, а это может произойти, если с новым размером построение чертежа станет невозможным, она выдаст сообщение о противоречии в данных. В этом случае пользователь должен интерактивно исправить свой проект.

Решение задачи поддержки целостности хорошо определенной модели можно представить как составление плана построения чертежа, начиная с некоторой фиксированной точки или линии. Определение геометрических свойств каждого следующего примитива производится на основе отношений, связывающих его с построенными ранее примитивами. Такой план можно построить, рассматривая степени свободы объектов модели. Обзор других подходов к решению данной задачи сделан в работе [15].

Обработка контекста ПИ. ПИ становится компактнее и понятнее пользователю, если его действия интерпретируются в соответствии с контекстом диалога. Под контекстом здесь понимается текущее состояние диалога и прикладной модели, представленные пользователю в графическом виде. Важность обращения к контексту покажем на примере диалога с чертежно-графической системой. Допустим, нам требуется разработать диалог для нанесения размеров на чертеже. Трудность этой задачи обусловлена разнообразием видов размеров. Они могут быть линейными, угловыми, радиальными. Каждый из названных видов имеет подвиды. Так, линейные размеры могут быть вертикальными, горизонтальными, между двумя точками, от точки до прямой, между двумя прямыми. Для каждого подвида возможно различное оформление размера: размерное число может располагаться посередине размерной линии, сбоку или на полке. Пользуясь общепринятым подходом, такое разнообразие вариантов можно обеспечить, разработав палитру инструментов, вызывающих соответствующие интерактивные методы нанесения размеров. Но в таком случае получается сложный в изучении и использовании ПИ.

Более удачное решение можно получить, если команда нанесения размера будет использовать контекст: пользователь введет начало и конец размерной линии, а система, исходя из контекста, выведет тип и некоторые геометрические параметры. Остальные параметры могут быть заданы при дальнейшем редактировании, например с использованием метафоры эволюционного развития.

Функцию обработки контекста можно возложить на систему вывода геометрических соотношений. Например, для задачи простановки размеров она, получив список гипотез о геометрических соотношениях, выведет тип размера и связанные с ним объекты.

Реализовать обработку контекста можно средствами чистого функционального программирования на языке Хаскель [16]. Этот выбор сделан по следующим причинам: во-первых, программа тогда будет математически точным определением, во-вторых, есть достаточно хорошо разработанные

методы доказательства свойств таких программ и, в-третьих, в последнее время разрабатываются методы, основанные на теории категорий и понятии монады, позволяющие реализовывать сложные вычислительные процессы.

В данной работе представлен метод обработки контекста ПИ, в котором процесс работы продукционных правил моделируется монадами [17]. Монады позволяют описывать последовательности действий в рамках функционального программирования. Для нашей задачи подходит монада **`Maybe`**, моделирующая действие, которое может дать результат, а может и не дать. Последовательностью таких действий реализуется продукционное правило системы логического вывода. Поскольку монада **`Maybe`** позволяет суммировать действия оператором **`mplus`**, соединив им реализации отдельных продукций можно получить систему логического вывода.

Задачу обработки контекста диалога рассмотрим на примере простановки размеров в чертежно-графической системе. Пусть пользователь ввел две точки, которые определяют отрезок. Эти точки и отрезок геометрически соотносятся с контекстом. Геометрические соотношения могут быть самыми разнообразными, приведем только те из них, которые будут использованы в данной статье.

Для точки с точкой:

1) введенная точка совпадает по координате X с некоторой точкой из контекста,
2) аналогичное совпадение по координате Y,
3) введенная точка полностью совпадает с некоторой точкой из контекста.

Для точки с прямой:

4) введенная точка лежит на прямой из контекста,
5) введенная точка принадлежит перпендикуляру, проведенному из конца отрезка из контекста.

Для отрезка:

6) введенный отрезок параллелен отрезку из контекста,
7) отрезок перпендикулярен отрезку из контекста.

Эти соотношения представим алгебраическим типом:

```
data Guess = EqX Int
           | EqY Int
           | EqPnt Int
           | PntOnLine Int                                    (1)
           | PntOnPerp Int Int
           | Parallel Int
           | Perpend Int
```

Здесь конструкторы данных следуют в том же порядке, что и перечисленные выше соотношения. Конструкторы соответственно содержат следующие значения:

1) индекс координаты X,
2) индекс координаты Y,

3) индекс точки,
4) индекс отрезка,
5) индексы отрезка и точки, через которую проведен перпендикуляр,
6) индекс отрезка,
7) индекс отрезка.

Будем считать, что для каждой введенной точки формируется список геометрических соотношений типа **`[Guess]`**, причем для второй точки могут быть соотношения с прямой, проходящей через две введенные точки. Итак, для двух введенных точек имеем два списка: **`g1::[Guess]`** и **`g2::[Guess]`**. Они будут использоваться системой продукций как факты. Продукционные правила будут искать факты и в случае успеха запускать на выполнение действия функционального ядра. В таблице приведено соответствие найденных соотношений действиям.

Таблица

Соответствие контекста выполняемому действию

№ варианта	Геометрические соотношения с контекстом	Выполняемое действие
1	Два совпадения по координате X	Вставка горизонтального линейного размера
2	Два совпадения по координате Y	Вставка вертикального линейного размера
3	Указаны две точки	Вставка линейного размера, определяющего расстояние между двумя точками
4	Для введенного отрезка существует параллельный отрезок. Введенные точки лежат на перпендикулярах к этому отрезку, проведенных из его концов	Вставка направленного линейного размера, определяющего расстояние между двумя точками, при этом учитывается расстояние от отрезка до размерной линии
5	Указаны точка и отрезок	Вставка линейного размера, определяющего расстояние от точки до прямой
6	Указаны два параллельных отрезка	Вставка линейного размера, определяющего расстояние между двумя параллельными прямыми

Поиск фактов будем вести стандартной функцией **`find`** которая, получив предикат **`p`**, задающий свойство искомого объекта и список объектов возвращает **`(Just x)`**, если **`x`** — первый объект в списке, удовлетворяющий свойству **`p`**, или в противном случае — **`Nothing`**. Например, поиск совпадений по координате **X** определяется функцией:

```
findEqX = find eqX
eqX (EqX _) = True                (2)
eqX   _ = True
```

Далее будем считать, что для каждого конструктора типа **Guess** у нас есть соответствующая функция поиска. Эти функции мы будем использовать для моделирования продукционных правил, которые реализуются на основе монад.

Например, правило для вставки горизонтального размера согласно определению, приведенному в таблице 1, кодируется следующим образом:

```
horDimRule = do
  (EqX ix1) <- findEqX g1
  (EqX ix2) <- findEqX g2                (3)
  constrHorDim p1 ix1 p2 ix2
```

Аналогично определяются и правило для вставки направленного размера:

```
parDimRule = do
  (Parallel ipar) <- findParallel g2
  (PntOnPerp ip1 il1) <- find PntOnPerp g1          (4)
  (PntOnPerp ip2 il2) <- find PntOnPerp g2
        constrParPPDim ipar ip1 ip2 il1 il2
```

Продукции объединяются в систему операцией mplus с учетом их приоритета. Более подробно вопросы реализации монадического взаимодействия с контекстом изложены в [18].

Рассмотренный метод обработки контекста ПИ позволяет декларативно описывать интерактивный процесс, за счет чего становится более понятной связь между руководством пользователя и программной реализацией. Кроме этого облегчается процесс модификации программы и таким образом появляется возможность поэтапного ее усовершенствования.

Выводы:

1. Показано, что разработка новых метафор диалога, таких как метафора эволюционного развития и метафора взаимодействующих объектов, может сделать ПИ более компактным и понятным.

2. Введение в ПИ системы поддержки ограничений целостности существенно расширяет возможности САПР.

3. Интерпретация действий пользователя в соответствии с контекстом диалога позволяет упростить ПИ и сделать его более понятным.

4. Применение функционального программирования практически сводит к нулю вероятность ошибок в программе и позволяет сконцентрировать внимание разработчика ПИ на прикладных аспектах.

Список литературы: 1. *Shneiderman B.* Direct manipulation. A step beyond programming languages // Computer.- 1983. - Vol.16. - N.8.-P.57-69 **2.** *Гордиенко А.П.* Процессы диалога // Известия ОрелГТУ. Серия „Информационные системы и технологии".- 2005.- N 2(8).- С. 50 -61. **3.** *Гордиенко А.П.* Язык взаимодействия объектов в пользовательском интерфейсе прямого манипулирования // Материалы конференции Графикон'94.- Нижний Новгород 1994. С.105-110. **4.** *Гордиенко А.П.* Разработка графического пользовательского интерфейса методами чистого функционального программирования // Материалы международной научной конференции „Пользовательский интерфейс в современных компьютерных системах".- Орел: Орловский государственный технический университет, 1999. - С. 98–103. **5.** *Sutherland I.E. Sketchpad: A* Man-Machine Graphical Communication System // Proceedings of the Spring Join Computer Conference.– 1963.- P.329-345. **6.** *Myers B. A., Richard G. McDaniel, Robert C. Miller, Alan S. Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski and Patrick Doane.* The Amulet Environment: New Models for Effective User Interface Software Development // IEEE Transactions on Software Engineering– 1997.– Vol.23,– N.6.– P.347-365. **7.** Nelson G. Juno, a constraint-based

graphics system // SIGGRAPH'85.– 1985.– Vol.19.– N.3.– P.235–243. **8.** *Borning A.,* Duisberg. Constraint-Based Tools for Building User Interfaces // ACM Transactions on Graphics.– 1986.– Vol.5.– N.4.– P.345-374. **9.** *Bass L., Coutaz J., Unger C.* A reference model for interactive system construction. In Gornostaev J. (Ed.) Proceedings of the East-West International Conference on Human-Computer Interaction EWHCI'92, St. Petersburg, Russia, 4-8 August, 1992, International Center for Scientific and Technological Information, Moscow, pp. 23-30. **10.** *Гордиенко А.П.* Модели графического пользовательского интерфейса // Вестник МЭИ.- 2003.- N 2.- С. 83-90. **11.** *Гордиенко А.П., Чижов А.В.* Объектно-ориентированная модель системы объемного проектирования изделий сложной формы // Инженерный журнал-справочник.– 1999.– N 1.- С.26-30. **12.** *ISO 7942-1985E.*- Information Processing System.- Computer Graphics.- Functional Specification of the Graphical Kernel System (GKS). **13.** *Гордиенко А.П.,* Построение графического пользовательского интерфейса в виде иерархии интеракторов // ИТНОП Известия ОрелГТУ Информационные системы и технологии, 2004, т. 2 (3). С. 110-116. **14.** *Хоар Ч.* Взаимодействующие последовательные процессы.- М: Мир, 1989. **15.** *Чижов А.В.* Использование ограничений в графическом редакторе, поддерживающем целостность геометрической модели // ИТНОП Известия ОрелГТУ Информационные системы и технологии, - 2004, т. 2 (3) С. 48-52. **16.** *Hudak, P.* Haskell 98 Language and Libraries. The Revised // Technical report. – Yale University and Glasgow University. – 2002. – 151 p. **17.** *Wadler, P.* Monads for functional programming //Advanced Functional Programming, Springer Verlag, LNCS 925. – 1995. С. 24-52. **18.** *Гордиенко А.П.* Основанная на монадах обработка контекста в графическом пользовательском интерфейсе // Материалы IV-й международной научно-технической конференции «Информационные технологии в науке, образовании и производстве» (ИТНОП).− Орел: ОрелГТУ, 2010, т. 3 С. 68-73.

Поступила в редколлегию 30.10.11