УДК 004.92

© 2016 г. **В.В. Грибова,** д-р техн. наук,
Л.А. Федорищев, канд. техн. наук
(Институт автоматики и процессов управления ДВО РАН, Владивосток)

УПРАВЛЕНИЕ ВИРТУАЛЬНЫМИ СЦЕНАМИ*

Рассмотрена концепция разработки и управления профессиональными виртуальными средами с помощью инструментального комплекса. Представлено управление логической моделью виртуальной среды при помощи новой расширенной онтологии. Описана технология управления графическим содержимым 3*D*-сцен. Рассмотрен процесс интерпретации виртуальной среды с учетом проверки ее целостности.
Ключевые слова: онтологии, управление, виртуальные среды.

Введение

В настоящее время при создании программного обеспечения большое внимание уделяется его жизнеспособности или максимально возможному увеличению жизненного цикла. Получены данные [1], что в процессе жизненного цикла программного средства около 30% всех усилий уходит на стадию разработки и более 70% – на стадию сопровождения в связи с изменениями требований пользователей, условий эксплуатации, предметной области, ошибками и дефектами в программном обеспечении [1, 2]. Известно, что сложность и комплексность программного обеспечения постоянно увеличиваются, поэтому большое внимание исследователей уделяется созданию методов и программных средств, направленных на снижение трудоемкости разработки, особенно сопровождения программного обеспечения.

Одним из больших классов задач, который в настоящее время востребован в бизнесе, образовании и науке, являются профессиональные виртуальные среды (3*D*-миры, виртуальная реальность) [3].

Особенностью профессиональных виртуальных сред является то, что они содержат предметные знания; их носителями являются эксперты предметной области, которые должны эти знания формировать и сопровождать. Вместе с тем на сегодняшний день существуют различные инструментальные средства, пакеты прикладных программ, библиотеки для создания виртуальных сред общего на-

* Работа выполнена при частичной финансовой поддержке РФФИ, № 15-07-03193 и ДВО РАН (проект «Дальний Восток»).

значения: Дельфин, *ToolBook, Lectora, CAVE, WorldToolKit, Avango, Lightning, Juggler, Unity3D, Virtools, Alternativa3D, Flare3D* и многие другие. Однако в настоящее время неизвестны специализированные средства для разработки именно профессиональных виртуальных сред (ПВС), между тем их разработка, особенно сопровождение с использованием универсальных инструментальных средств, является исключительно трудоемким процессом.

Авторами была предложена концепция разработки ПВС на основе знаний, с разным уровнем детализации описанная в работах [4 – 7], реализован программный комплекс, с использованием которого были созданы прикладные облачные сервисы для решения различных задач [4]. Использование программного комплекса, с одной стороны, показало перспективность предложенного подхода, значительное снижение трудоемкости разработки и сопровождения виртуальных сред, анализ которой дан в работе [7], с другой стороны, стали очевидными ограничения предложенной модели виртуальной среды, а также выделены дополнительные механизмы, направленные на повышение управляемости профессиональных виртуальных сред в процессе их жизненного цикла.

Данная работа описывает расширенную концепцию инструментария для создания, функционирования и управления ПВС, основная цель которой – обеспечение жизнеспособности ПВС различного типа и назначения.

1. Основные принципы разработки инструментария

Главная цель создания специализированного инструментария – упростить создание ПВС и управление ими в процессе жизненного цикла. Напомним, что под управлением понимается решение задач сопровождения программного средства с помощью специальных высокоуровневых механизмов управления, сводящих к минимуму изменение его кода [8, 9].

Основу ПВС составляют предметные знания экспертов предметной области. Таким образом, данный тип инструментария можно отнести к классу интеллектуальных систем. В архитектуре интеллектуальных систем выделяется дополнительная компонента – база знаний, и в соответствии с современным подходом она разрабатывается и сопровождается экспертами предметной области [6, 9]. В данном инструментарии также предлагается выделить знания экспертов в отдельную компоненту – логическое представление, разработать унифицированную онтологию логического представления виртуальной среды и создать программное средство (редактор) для управления экспертными знаниями (формирование логического представления виртуальной среды по онтологии).

Не менее существенной частью системы является непосредственно визуальное 3*D*-представление виртуальной среды (виртуального мира), который основан на логическом представлении, сформированном экспертами, и должен разрабатываться графическими дизайнерами. Таким образом, в инструментарии необходимо предусмотреть «рабочее место» графического дизайнера.

Любая логика работы может иметь вспомогательный набор функций обработки, связанный как с элементами графики, так и с обработкой логических параметров виртуальной среды.

Наконец, любая виртуальная среда помещена в некоторое «функциональное окружение», т.е. встраивается в пользовательский интерфейс клиентского приложения или web-браузер. Пользовательский интерфейс содержит вызов некоторых дополнительных или сервисных функций, которые определяются типом или назначением профессиональной виртуальной среды. Такими функциями являются, например, запуск или остановка виртуальной среды, сохранение или задание необходимых параметров, команд, которые сложно (невозможно или неудобно) выполнить в виртуальной среде. Разработку пользовательского интерфейса осуществляют дизайнеры.

Для упрощения разработки и прежде всего сопровождения ПВС предлагается заменить кодирование на языке программирования проектированием и управлением ее декларативной модели с проверкой на целостность и непротиворечивость, а также интерпретацией этой модели вместо традиционной компиляции с последующим запуском. При этом саму модель разделить на компоненты в соответствии с типами разработчиков и предоставить каждому типу разработчиков систему понятий, в терминах которой он будет разрабатывать и сопровождать свою часть модели ПВС. Дополнительно к декларативной модели добавить возможность включения в нее вычислительных функций, реализованных программистами, для расширения функциональных возможностей ПВС.

Следующим, принципиально важным требованием является использование как средства разработки ПВС, так и готового приложения как облачных сервисов. Разработка ПВС и использование готового приложения как облачных сервисов не требуют их установки на компьютере пользователя или разработчика, не предъявляют никаких дополнительных условий к операционной системе, оперативной памяти и другим техническим требованиям их компьютеров. Очень важно использование данной технологии для обеспечения жизнеспособности готового приложения, поскольку в процессе всего жизненного цикла модель ПВС доступна для сопровождения. Практически только эта технология и позволяет обеспечить управление ПВС в процессе ее использования. Удаленное использование позволяет не только расширить аудиторию пользователей средства, но также и привлекать разработчиков независимо от их географического расположения.

В качестве платформы для реализации ПВС выбрана платформа *IACPaaS* [10]. Облачная платформа *IACPaaS* представляет собой программно-информационный интернет-комплекс для обеспечения поддержки разработки, управления и удаленного использования прикладных и инструментальных мультиагентных облачных сервисов (прежде всего интеллектуальных) и их компонентов. Комплекс основан на технологии облачных вычислений и обеспечивает удаленный доступ конечным пользователям к интеллектуальным системам, а разработчикам и управляющим – к средствам создания интеллектуальных систем и управления ими.

2. Управление логическим представлением модели ПВС

Логическое представление ПВС является ключевым описанием, по которому, с одной стороны, формируется визуальное представление виртуальной среды,

с другой стороны, оно определяет логику ее работы. Логическое представление включает описание объектов виртуальной среды и их свойств (атрибутов), возможные действия пользователя в виртуальной среде и результаты, к которым могут привести эти действия. Вся информация, составляющая логическое представление модели ПВС, является знаниями экспертов предметной области, соответственно эксперты должны не только формировать их, но также и сопровождать в процессе жизненного цикла ПВС.

Современный подход к системам, основанным на знаниях, базируется на двухуровневом принципе создания баз знаний [5, 6]: на первоначальном этапе создается онтология предметной области, затем по ней эксперт формирует и сопровождает базу знаний. Анализ ПВС и их функциональных возможностей показал, что можно разработать универсальную онтологию ПВС, не зависящую от предметных областей и ориентированную на типичные задачи, которые решаются в системах такого класса (см. раздел 5) и далее на основе этой онтологии формировать и сопровождать модель ПВС, заменив кодирование на языке декларативным высокоуровневым описанием логики ПВС. Такая онтология была создана [6], на основе нее разработан соответствующий инструментарий, который показал применимость и перспективность данного подхода.

Однако опыт использования инструментария свидетельствует, что онтология имеет ограничения, которые сужают либо усложняют применение инструментария для задач моделирования. Особенностью такого класса задач является то, что на этапе проектирования невозможно задать все описания объектов и их атрибутов. Объекты виртуальной среды могут динамически генерироваться либо пользователями, либо программным средством, обеспечивающим логику функционирования, поэтому на этапе проектирования возможно задать только класс объекта и описать множество его логических атрибутов, не фиксируя сами объекты. С учетом вышесказанного, разработана усовершенствованная онтология профессиональной виртуальной среды, которая принимает во внимание особенности задач моделирования.

Онтология ПВС состоит из трех основных компонентов: объектов виртуальной среды, действий, которые можно выполнить с объектами виртуальной среды, и сценария: *Онтология = <Объекты, Действия, Сценарий>* (при описании онтологии используется теоретико-множественная нотация).

Объекты виртуальной среды разделены на следующие классы: простой (неизменяемый) объект, изменяемый объект, составной объект, таблица: *Объекты ∈ {Простой объект, Изменяемый объект, Составной объект, Таблица}*.

Простой объект – это объект, обладающий набором атрибутов, которые не изменяются в процессе воспроизведения сцены. *Простой объект = <Характеристика объекта, Имя класса, Имя объекта, Описание, Атрибуты>; Характеристика объекта = {Класс, Экземпляр}*.

Атрибуты = <Логические атрибуты, Презентационные атрибуты>.

Эксперт предметной области создает и управляет в процессе жизненного цикла множеством логических атрибутов. Множество презентационных атрибутов проектируется графическим дизайнером интерфейса (см. раздел 3).

Логические атрибуты = {*Логический атрибут*}$_{i=0}^{n}$. *Логический атрибут* = <*Имя* ∈ *Строка, Значение*∈*Типы данных*>; *Типы данных* = <*Простые типы, Составные типы*>.

Простые типы ∈ {*Целое число, Вещественное число, Строка, Логическое значение*}.

Составные типы ∈ {*Множество, Вектор, Интервал целых чисел, Интервал вещественных чисел, Вычисляемое значение*}.

Изменяемый объект – это объект, который является расширением простого объекта, а также включает множество изменяемых атрибутов, которые могут быть самостоятельными или объединены в логические группы – состояния. Каждое состояние определяется совокупностью значений атрибутов, которые они могут принимать в ответ на действия, происходящие в виртуальной среде. *Изменяемый объект* = <*Характеристика объекта, Имя класса, Имя объекта, Описание, Логические атрибуты, Презентационные атрибуты, Множество состояний изменяемого объекта, Начальное состояние изменяемого объекта*>.

Действия в онтологии виртуальной среды описывают все действия, которые можно производить с объектами виртуальной среды либо с классами объектов: *Действия* = {*Действие*$_i$ | *Действие*$_i$ ∈ *Действие*}$_{i=1}^{n}$.

Действие = <*Имя действия, Описание, Способы выполнения, Входные параметры, Изменение состояния объектов, Получение оценки, Параметры обработки, Сообщение*>

Способы выполнения ∈ {<*Интерактивные, Командное, Простое*>}.

Выделяются следующие способы выполнения действий: *командные действия* – это действия, которые пользователь совершает над объектами либо классами объектов вне пространства виртуальной среды (с помощью элементов интерфейса, голоса); *интерактивные действия* – это действия, которые пользователь совершает над объектами либо классами объектов, находясь в пространстве виртуальной среды; *простые действия* – это действия, которые пользователь не выполняет непосредственно, они выполняются виртуальной средой на основе параметра другого действия.

Сценарий предназначен для описания последовательности действий, которые должен выполнить пользователь для получения результата, определяемого обучающим заданием. С одной стороны, сценарий представляет собой граф, в вершинах которого находятся действия, а дуги являются переходами от одних действий к другим. С другой стороны, сценарий – это последовательность логических этапов, где каждый этап представляет собой объединенное в группу множество вершин. *Этапы* могут иметь параметры, уточняющие особенности его выполнения. *Вершины* графа представляют собой ожидаемые от пользователя действия, связанные *дугами* (*переходами*). *Переходы* могут быть: *фиксированные* – это постоянные переходы, которые не имеют зависимости от результата; *по результату* –переходы, которые зависят от результата действия, имеют несколько взаимоисключающих вариантов переходов; *с условием* –переходы, которые выполняются только при наличии выполненного условия, независимо от конкрет-

ных результатов предыдущего действия.

Сценарий = <Имя сценария, Этапы, Последовательность этапов>, Имя сценария∈ Строка, Этапы = { Этап } $_{i=0}^{n}$.

Этап = <Входные параметры, Вершины, Дуги, Начальная вершина, Конечная вершина>.

Вершины = {Вершина$_i$*∈ Вершина }* $_{i=1}^{n}$, *Дуги = {Дуга*$_i$*∈ Дуга}* $_{i=1}^{n}$, *Переход ∈ {Простой, По оценке, По условию}*.

Логическое описание онтологии ПВС подается на вход редактору *IWE*, который является составным компонентом платформы *IACPaaS*. По описанию онтологии редактор автоматически генерирует интерфейс пользователя – эксперта предметной области, который формирует логическое представление модели ПВС в терминах онтологии (рис. 1).

Рис. 1. Редактор логического представления модели виртуальных сред.

3. Управление 3*D*-сценами

На основе сформированного экспертами предметной области логического представления графический дизайнер должен сформировать 3*D*-сцену. Формирование графического представления осуществляется по онтологии графического представления, являющегося расширением онтологии логического представления. Задача дизайнера – сопоставить логическим атрибутам объектов множество их презентационных атрибутов. В онтологии ПВС выделены следующие презентационные атрибуты: *Презентационные атрибуты = <Модель, Текстуры, Координаты, Повороты, Масштабы, Анимация >*.

Модель – это 3*D*-модель (мэш), которая используется для создания трехмерного образа объекта. *Текстуры* –растры, используемые для покрытия 3*D*-модели. *Координаты, Повороты и Масштабы* –тройки вещественных чисел, оп-

ределяющие объект в пространстве. *Анимация* – название анимации, которая должна быть воспроизведена для данного объекта.

Таким образом, для каждого объекта виртуальной сцены должны быть определены все презентационные атрибуты. Задание данных атрибутов осуществляется с помощью специализированных инструментов графического редактора (рис. 2). Для этого сформированное экспертом логическое представление загружается в графический редактор, и дизайнер визуальными средствами описывает презентационные атрибуты.

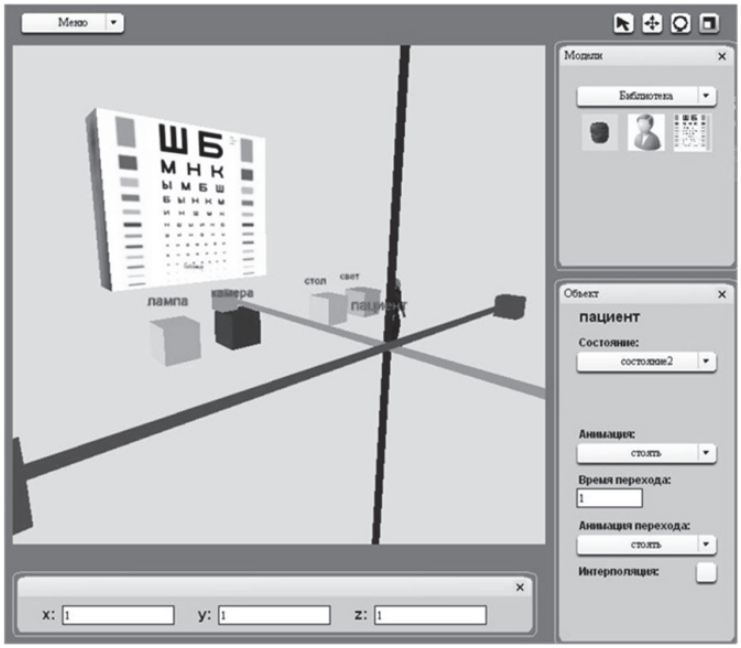

Рис. 2. Интерфейс графического редактора.

Помимо задания презентационных атрибутов объектов, в онтологии введены два вспомогательных типа объектов – источник света и камера. *Источник света* – это служебный объект, не имеющий собственного отображения и служащий для указания в пространстве сцены точки и направления излучения света, а также визуальных характеристик света. Источник света наследуется от изменяемого объекта, так как его атрибуты могут быть динамически изменены. *Камера* – это служебный объект, который, как и источник света, не имеет собственного отображения, он служит для указания в пространстве сцены точки, в которой находится пользователь. Объект "камера" наследуется от простого объекта.

4. Управление созданием специфических функций обработки

В описании декларативной модели ПВС могут понадобиться параметры, которые должны вычисляться специфическим (нестандартным) способом. Поэтому к декларативному описанию модели ПВС, сформированному экспертом предметной области и графическим дизайнером, могут в случае необходимости (но не обязательно) быть добавлены дополнительные функции обработки. В этом случае такой параметр помечается как "параметр обработки агентом", для него в модели указывается ссылка на функцию, реализующую эту обработку. Платформа *IACPaaS* реализована как мультиагентная, поэтому эти функции реализуются как

агенты платформы. Агент состоит из двух частей – декларативной и процедурной. Декларативная часть состоит из документации к агенту и формальной спецификации множества блоков продукций, из которых состоит агент [10]. Платформа *IACPaaS* поддерживает автоматизацию процесса разработки и сопровождения документированного исходного кода агента. Орграфовая связная двухуровневая модель информационных ресурсов и ее соответствующая поддержка процессором информационных ресурсов платформы *IACPaaS* позволяют хранить декларативную спецификацию и исполняемый код агента (получаемый в результате компиляции его исходного кода) в одной единице хранения фонда, представляющей данный агент. Для создания и обработки таких единиц хранения платформа предоставляет инструментальный сервис [10].

Процедура генерации заготовки исходного кода разрабатываемого агента, в которую встраивается документация к последнему, позволяет существенно упростить разработку и сопровождение процедурной части агента.

Разработка агента состоит в формировании в фонде платформы *IACPaaS* информационного ресурса, представляющего декларативную спецификацию агента; генерации заготовки исходного кода агента по его декларативному описанию; написании исходного кода агента (в частности, кода блоков продукций агента); получении байт-кода агента (в результате компиляции исходного кода) и загрузки его в фонд – в соответствующий информационный ресурс [10]. Платформа также позволяет обращаться из агента к внешним *web*-серверам для получения и передачи данных. Эта функция необходима, когда вычисление значений некоторых параметров виртуальной среды производится удаленно, – например, на высокопроизводительных вычислительных архитектурах.

5. Управление *WIMP*-интерфейсом

Принципиально новым расширением в разработке и управлении ПВС стала возможность декларативного формирования *WIMP*-интерфейса, здесь *WIMP* – *Windows, Icons, Menu, Pointing devices*. Их анализ показал, что независимо от предметной области и решаемых задач, где используются виртуальные среды, их все можно разделить на несколько классов, каждый из которых имеет фиксированный основной функционал и, возможно, некоторый дополнительный. Набор фиксированных функций определен в интерпретаторе, поэтому не требует дополнительного программирования и обработки. Однако в случае включения дополнительных функций программист может их добавить (см. раздел 4) и, если необходимо, описать соответствующий *WIMP*-интерфейс.

Онтология *WIMP*-интерфейса ПВС предназначена для упрощения создания и сопровождения *WIMP*-интерфейса, который является “окружением” виртуальной среды.

Онтология имеет следующий вид: *WIMP-интерфейс* = {*Задача* $_i \in$ *Тип задачи*}, 0 <= *i* <= количество задач;

Тип задачи $\in$ {*Стандартная задача, Расширенная задача*}.

Стандартная задача $\in$ {*Проигрыватель, Моделирование, Обучение, Обучение с контролем*}.

Задача$_i$ = {<[*Имя подзадачи*]$_{ij}$, *Представление*$_{ij}$ >| *Задача* $_i$ ∈ *Стандартная задача*}.

Задача$_i$ = {<[*Имя подзадачи*]$_{ij}$, *Агент обработки*$_{ij}$, *Представление*$_{ij}$>| *Задача*$_i$ ∈ *Расширенная задача*}.

Представление $_{ij}$ = <[*Задача* $_{ijk}$]{*Элемент интерфейса*}$_{jk}$ >.

Элемент интерфейса$_i$ =< *Параметры*$_i$,*События*$_i$ >.

Параметры$_i$ = {*Параметр*$_{ij}$}$_{j=0}^{n}$, *События*$_i$ – множество событий, *События*$_i$ = {*Событие*$_{ij}$}$_{j=0}^{n}$ $^{j=0}$ ^{eventcount} .

Каждый параметр элемента интерфейса описывается своим типом и значением, т.е. *Параметр*$_i$ = <*Значение параметра, Тип параметра*$_i$>.

Тип параметра элемента интерфейса *Тип параметра*$_i$ ⊂ *Типы параметров,* где *Типы параметров* = *Строка* ∪ *Целое* ∪ *Вещественное* ∪ *Логическое* ∪ *Изображение* ∪ *Входной параметр* ∪ *Выходной параметр.*

События элемента интерфейса *События*$_i$ = {*Событие*$_{ij}$}$_{j=0}^{n}$ задают множество тех событий, на которые элемент интерфейса может реагировать.

Каждое событие *Событие*$_{ij}$ описывается следующим образом: *Событие*$_{ij}$ = <*Имя события*$_{ij}$, *Параметры события*$_{ij}$>.

Множество *Параметры события*$_{ij}$ = {*Параметр события*$_{ijk}$}$_{k=0}^{n}$. Каждый параметр задается своим именем и типом: *Параметр события*$_{ijk}$ = <*Имя параметра*$_{ijk}$, *Тип параметра*$_{ijk}$>, где *Тип параметра* ∈ *Типы параметра. Типы параметра* = *Строка* ∪ *Целое* ∪ *Вещественное* ∪ *Логическое.*

6. Целостность модели и запуск виртуальной среды

Проверка целостности разработанной декларативной модели профессиональной виртуальной среды (возможно, с включением в нее дополнительных агентов обработки) состоит из двух уровней. Первый уровень заключается в том, что сама модель формируется на основе онтологии, содержащей набор ограничений целостности, который проверяется автоматически при создании каждой компоненты декларативной модели. Второй уровень проверки целостности осуществляется на программном уровне в графическом редакторе и интерпретаторе, в каждом из которых проверяется наличие всех необходимых на данный момент сущностей декларативной модели для воспроизведения, а также их корректность, т.е. наличие необходимых ресурсов, ссылок, отсутствие перекрестных ссылок и т.п. Таким образом, второй уровень проверки целостности обеспечивает частичную семантическую проверку целостности декларативной модели.

Интерпретатор предназначен для запуска и работы созданных профессиональных виртуальных сред и состоит из двух частей: серверного приложения, размещенного на платформе, и веб-клиента, с которым работает пользователь. Каждое разработанное прикладное приложение становится отдельным сервисом облачной платформы. На рис. 3 приведена общая архитектура системы, демонстрирующая все основные блоки и связь между компонентами модели системы.

Обсуждение

Известно, что создание интеллектуальных систем, их сопровождение сопряжено с большой трудоемкостью и требует участия специалистов разной профессиональной подготовки – инженеров по знаниям для создания онтологии предметной области, экспертов предметной области для разработки и сопровождения базы знаний, программистов и т.п. Напомним, что рассматривается современный двухуровневый подход к созданию баз знаний [5, 6].

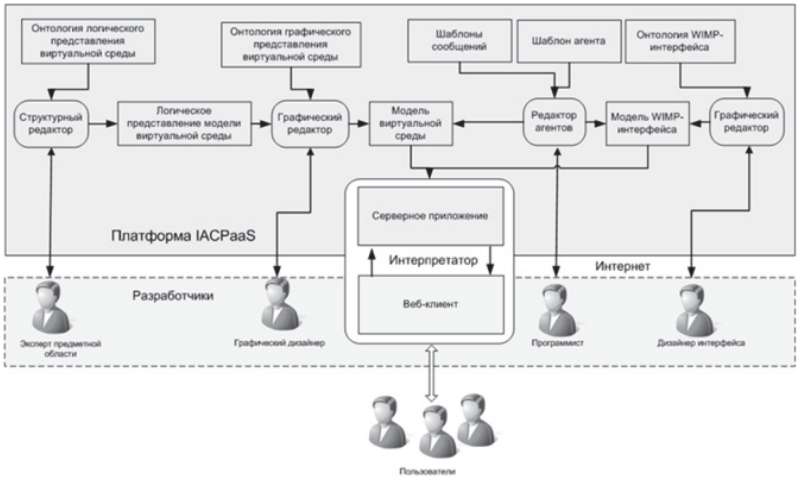

Рис. 3. Архитектура инструментального средства.

Создание виртуальных миров (виртуальных сред) является отдельной и также очень трудоемкой работой. Для того, чтобы их создание было экономически оправдано, необходимо, чтобы они, с одной стороны, имели как можно более длинный жизненный цикл, а с другой стороны, чтобы эти программные средства были ориентированы на как можно большую аудиторию пользователей. Проблема обеспечения жизнеспособности, как указывается в [8, 9], решается наличием механизмов управления программным средством (решение задач сопровождения программного средства с помощью специальных высокоуровневых механизмов, сводящих к минимуму изменение его кода).

Обсудим, какие механизмы, предложенные авторами, направлены на решение проблемы управления ПВС.

Анализ виртуальных сред позволил создать универсальную онтологию виртуальной среды, состоящую из двух основных частей: логического и графического представлений. Онтология является управляющей структурой, на основе которой эксперты предметной области и графические дизайнеры формируют декларативную модель ПВС. Формирование декларативной модели по онтологии осуществляется с помощью структурного и графического редакторов соответственно. Логическое представление является управляющей структурой, содержащей описание всех наборов данных и знаний, необходимых для функционирования виртуальной среды. Графическое представление – 3*D*-отображение логического представления. Проектирование *WIMP*-интерфейса, как неотъемлемой части любого приложения, также осуществляется дизайнером интерфейса по соответствующей онтологии с помощью графического редактора, при этом дизайнер выбирает типы задач, которые решает ПВС, а далее в соответствии с этими типами он

может либо выбрать удобное представление, либо в декларативном виде его описать. Далее интерпретатор ПВС проверяет модель на целостность и запускает ее на выполнение. Итак, проектирование ПВС и ее дальнейшее кодирование заменяется проектированием декларативной модели по онтологии с последующим управлением ею в процессе жизненного цикла с помощью редакторов модели.

В идеологии создания ПВС допускается, что может возникнуть необходимость создания дополнительных функций (программ на языке программирования), которые требуются для обработки некоторых параметров. Их реализация осуществляется по технологии, предложенной в платформе *IACPaaS*, которая упрощает создание функций-агентов за счет наличия в платформе инструментов генерации их "заготовок".

Второй важный аспект жизнеспособности программного средства, помимо управляемости, достигается, что уже отмечалось, через доступность программного средства как можно большей аудитории пользователей. Этот аспект достигается использованием платформы *IACPaaS*, которая реализует как инструментарий для разработки ПВС, так и готовую ПВС как облачные сервисы.

ЛИТЕРАТУРА

1. Ogheneovo E.E. Software Maintenance and Evolution: The Implication for Software Development // Journal of Industrial and Academic Research . – 2013. – Vol.7, No. 1.
2. *Jones C.* The Economics Of Software Maintenance In The Twenty First Century. – 2006. URL: http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf. (дата обращения: 15.05.2015).
3. *Князева Г.В.* Виртуальная реальность и профессиональные технологии визуализации // Вестник Волжского университета им. В.Н. Татищева. – 2010. – №15.
4. *Грибова В.В., Петряева М.В., Федорищев Л.А., Черняховская М.Ю.* Модель виртуального мира мультимедиа тренажера для медицинского образования // International Book Series "Information Science and Computing". Book 22. Applicable Information Models. -Sofia, Bulgaria: ITHEA. – 2011. – №22. – P. 140-148.
5. *Грибова В.В., Федорищев Л.А.* Обучающие виртуальные системы и средства их создания // Вестник информационных и компьютерных технологий. – 2012. – №3. – С. 48-51.
6. *Грибова В.В., Федорищев Л.А.* Разработка виртуальных интерактивных сред на основе онтологического подхода // Тр. 13-й нац. конф. по искусственному интеллекту с международным участием КИИ-2012. – Т.2. – Белгород: Изд-во БГТУ, 2012. – С. 144-151.
7. *Грибова В.В., Федорищев Л.А.* Интернет-комплекс для создания обучающих систем с виртуальной реальностью // Дистанционное и виртуальное обучение. – 2012. – № 7. – С.4-12.
8. *Norvig P., Cohn D.* Adaptive software. URL: http://norvig.com/adapaper-pcai.html (дата обращения: 18.05.2015).
9. *Грибова В.В., Клещев А.С., Шалфеева Е.А.* Управление интеллектуальными системами // Известия РАН. Теории и системы управления. – 2010. – №6. – С.122-137.
10. Грибова В.В., Клещев А.С., Крылов Д.А. и др. Проект *IACPaaS* – развиваемый комплекс для разработки, управления и использования интеллектуальных систем // Искусственный интеллект и принятие решений. – 2011. – №1. – С.27-35.

Статья представлена к публикации членом редколлегии А.С. Клещевымым.

E-mail:
Грибова Валерия Викторовна – gribova@iacp.dvo.ru;
Федорищев Леонид Александрович – fleo1987@mail.ru.