УДК 004.414.38

Н.А. Годовиченко

ИСПОЛЬЗОВАНИЕ ОНТОЛОГИЙ ПРИ РЕШЕНИИ ПРОБЛЕМЫ СОГЛАСОВАНИЯ СПЕЦИФИКАЦИИ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ ИНФОРМАЦИОННЫХ СИСТЕМ

Робота присвячена використанню онтологічного апарату для узгодження специфікації вимог до програмного забезпечення інформаційних систем. Запропонована методика узгодження вимог на базі використання онтологій як моделі представлення вимог до програмного забезпечення. Запропонована онтологічна модель представлення вимог. Запропонован алгоритм узгодження вимог на базі матриці правил узгодження та таблиці зовнішньої узгодженості.

Ключові слова: онтологія, програмне забезпечення, узгодження вимог, правила узгодженості.

Введение

Тенденции развития информационных систем связаны со всё более возрастающими требованиями к программному обеспечению этих систем. Разработка программного обеспечения, как правило, состоит из нескольких этапов: анализа требований, спецификации требований, проектировании, кодировании, тестировании и сопровождении программного обеспечения.

Улучшение качества программного обеспечения остается критической проблемой в процессе разработки и эксплуатации информационных систем. Более 70% выпущенных программных продуктов не соответствует изначальным требованиям заказчика, разрабатывается с опозданием и превышает бюджет разработки. Более 94% проектов по разработке программного обеспечения подвергаются полной переработке. Более чем в половине случаев причиной неудач является неадекватная, несогласованная или неточная спецификация требований [1].

Моделирование требований на этапе спецификации требований

Этап спецификации требований связан с доскональным моделированием требований заказчиков, определенных в процессе анализа требований. В качестве входной информации выступают неформальные требования заказчиков, а результатом этого процесса является выработка моделей спецификации проектных конструкций: модели

состояний, модели поведения и модели изменения состояний. Эти модели дают более формальное определение различных сторон (представлений) системы и реализуются в виде диаграмм на языке визуального моделирования.

Существует много языков визуального моделирования, которые могут представить проект с различных ракурсов. Наиболее часто используемым языком является язык UML (Unified Modeling Language) [2], который де-факто является стандартным средством для моделирования программных систем. Большим преимуществом при использовании UML является обширная коллекция различных нотаций, что дает разработчикам большую свободу спецификации программных систем.

Однако такая гибкость часто ведет к появлению несогласованностей в программном проекте. К сожалению, UML не имеет точно определенной семантики, чтобы четко определять и проверять согласованность модели.

В данной работе, под несогласованностью подразумевается ситуация, когда существует конфликт, разногласие или различные трактовки одного и того же или многих фактов, поведений или ограничений.

Одним из путей решения данной проблемы является совмещение различных представлений программного проекта в одну метамодель с точной семантикой, что позволяет применять к данной модели различные методы формальной проверки согласованности спецификации требований.

В качестве метамодели использовались различные языки формальной спецификации требований (нотация Z, а также ее расширения Object-Z и Z++, нотация AMN и другие), промежуточные математические нотации (сети Петри, автомат Крипке и другие) и другие методы формальной записи требований.

В данной работе предлагается использовать онтологический метод в качестве метода формализации семантики UML, а в качестве метамодели представления требований использовать онтологию.

Онтологический аппарат

В последнее время, онтологии стали широко применяться в различных сферах информационных технологий, таких как мультиагентные системы, обработка естественного языка, управление знаниями,

 ISSN 1562-9945

интеллектуальный анализ данных, создание цифровых библиотек, электронная коммерция и т.д.

Онтология – это база знаний специального вида, которая содержит семантическую информацию определенной предметной области. Компоненты онтологии могут быть разными, но практически все модели онтологий содержат определенные концепты (понятия, классы), свойства концептов (атрибуты, роли), отношения между концептами (зависимости, функции) и дополнительные ограничения, которые накладываются аксиомами.

На формальном уровне онтология – это система, которая состоит из наборов понятий и утверждений про эти понятия, на основе которых строятся классы, объекты, отношения, функции и теории [3]. Формально онтология определяется как:

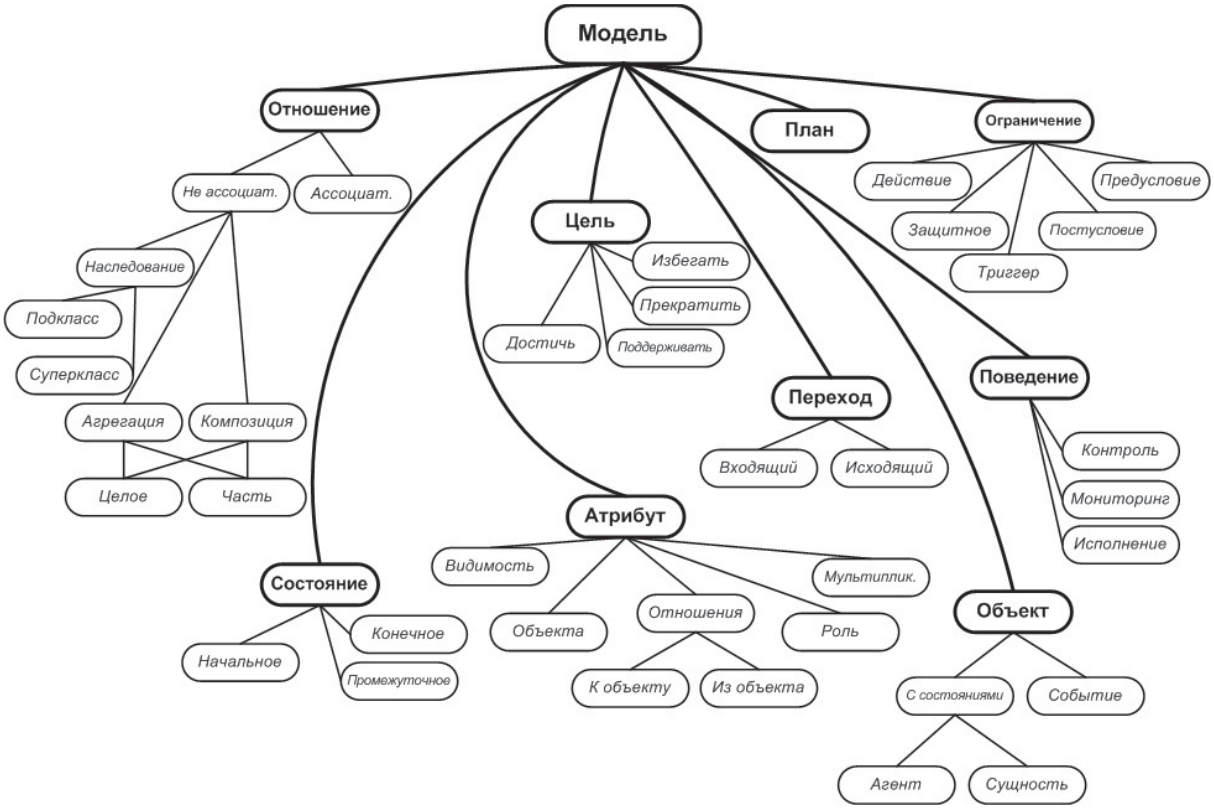

Рисунок 1 – Онтологическая модель представления данных (без указания свойств)

Свойства модели отображают отношения между концептами этой модели. В данной модели свойства реализуют отношение «иметь» (has), если не указано другое название отношения.

Процесс составления экземпляра модели заключается в последовательной обработке диаграмм UML, используемых на этапе спецификации требований, и компоновке экземпляра модели.

На этапе спецификации требований используются четыре диаграммы UML: диаграмма прецедентов (соответствует целям, которым должна соответствовать система), диаграмма классов (соответствует модели состояний системы), диаграмма последовательностей (соответствует модели поведения системы) и диаграмма автомата (соответствует модели изменения состояний).

При обработке UML диаграмм, их элементам ставятся в соответствие концепты онтологической модели (табл. 1).

Таблица 1

Соответствие диаграмм UML и концептов модели

Диаграмма UML	Концепты моделей
Диаграмма прецедентов	Цель
Диаграмма классов	Объект, атрибут, отношение, поведение
Диаграмма последовательностей	Поведение, план, состояние
Диаграмма автомата	Состояние, переход, ограничение

Структура модели обеспечивает представление системы с различных сторон и семантическое перекрытие различных представлений системы.

Диаграммы UML обрабатываются в следующем порядке:

1. диаграмма классов. На этом шаге определяются «объекты», «атрибуты», «отношения» и «поведения»;

2. диаграмма последовательностей. На этом шаге уточняются «поведения», а также определяются «ограничения», связанные с «поведением»;

3. диаграмма автомата. На этом шаге уточняются «ограничения» и определяются «состояния» и «переходы»;

4. диаграмма прецедентов. На этом шаге определяются «цели», связанные с «объектами» и «поведениями».

После обработки диаграмм происходит компоновка экземпляра модели, которая состоит в соединении набора разрозненных концептов в единую модель.

 ISSN 1562-9945

Алгоритм согласования требований

Согласование требований происходит после составления модели и базируется на использовании т.н. «правил согласования». Эти правила представлены в виде онтологических аксиом, которые являются ограничениями онтологий.

В данной работе представлены два вида правил: правила внешней согласованности и правила внутренней согласованности.

Правила внешней согласованности осуществляют проверку диаграмм в целом, тогда как правила внутренней согласованности осуществляют проверку внутри одной диаграммы или между двумя диаграммами.

Проверка согласованности в целом осуществляется путем составления таблицы внешней согласованности и применения к ней соответствующих правил.

Таблица внешней согласованности заполняется на этапе построения экземпляра модели, после обработки каждой диаграммы. В эту таблицу заносятся данные о наличии того или иного элемента в различных UML диаграммах.

Рассмотрим внешнее согласование на конкретном примере.

Введем правило внешнего согласования: «объект» должен быть определен в диаграмме классов и на него должна быть ссылка, по крайней мере, в одной диаграмме последовательностей и в одной диаграмме автомата:

$$\sqrt{\Omega} = \Xi\forall o1[object(o1) \rightarrow (class\_diagram(o1) \wedge (\exists o2[sequence\_diagram(o2) \wedge state\_diagram(o2)) \wedge o1 = o2]))]. \quad (1)$$

Заполним таблицу внешней согласованности (табл. 2).

Таблица 2

Пример таблицы внешней согласованности

Элемент UML	Концепт модели	Диаграмма классов	Диаграмма последовательностей	Диаграмма автомата	Диаграмма прецедентов
[class] Door	[oject:entity] Door	TRUE	FALSE	TRUE	FALSE

Как видно из таблицы, данное правило не выполняется, т.к. в диаграмме последовательностей данного объекта нет. Следовательно, обнаружена внешняя несогласованность элемента модели.

Важным вопросом при решении проблемы согласования требований является классификация несогласованностей. Введение классификации позволяет идентифицировать ошибку в диаграммах, которая привела к несогласованности и ускорить процесс её исправления. В данной работе предложена т.н. «матрица правил согласования», которая классифицирует правила в зависимости от концептов, которые принимают участие в том или ином взаимодействии (табл. 3).

Таблица 3

Структура «матрицы правил согласования»

	Объект	Атрибут	Поведение	Цель	Отношение	Состояние	Переход	Ограниче-ние
Объект	Об:об	Об:Ат	Об:Пов	Об:Цл	Об:От	Об:Сст	Об:Пх	Об:Огр
Атрибут		Ат:Ат	Ат:Пов	Ат:Цл	Ат:От	Ат:Сст	Ат:Пх	Ат:Огр
Поведение			Пов:Пов	Пов:Цл	Пов:От	Пов:Сст	Пов:Пх	Пов:Огр
Цель				Цл:Цл	Цл:От	Цл:Сст	Цл:Пх	Цл:Огр
Отношение					От:От	От:Сст	От:Пх	От:Огр
Состояние						Сст:Сст	Сст:Пх	Сст:Огр
Переход							Пх:Пх	Пх:Огр
Ограничение								Огр:Огр

На основе этой матрицы строятся правила определения внутренней несогласованности. Эти правила затрагивают взаимодействие между двумя концептами и находятся на пересечении строк и столбцов матрицы. Например, правило затрагивающее взаимодействие концептов «Объект» и «Поведение» называется Об:Пов. Приведем пример правила Об:Пов1: сообщение, посланное из «Объекта» должно быть связано с «Поведением» этого «Объекта»:

 ISSN 1562-9945

$$\forall o1 \exists o2[(object : statebased(o1) \wedge object : statebased(o2)) \rightarrow (\exists b(behavior(b) \wedge (property\_has(o1, b) \wedge ((property\_sends\_message\_to(b, o2) \vee \exists e(object : event(e) \wedge (property\_causes(b, e) \wedge property\_sends\_message\_to(e, o1))]))]))]. \quad (2)$$

Данное правило является ограничением для онтологии. В случае если экземпляр модели выходит за рамки наложенных ограничений, полученная онтология будет противоречива, что выявляется существующими средствами формальной логики.

Заключение

Проведенные экспериментальные исследования показали, что применение онтологии в качестве модели представления данных позволило сократить время на обработку моделей спецификации требований примерно на 7%, однако существенным преимуществом онтологического аппарата оказалось легкость в освоении методики и прозрачность процесса преобразования UML диаграмм в онтологическую модель.

ЛИТЕРАТУРА

1. Glib, T. What`s wrong with requirements specification? An analysis of the fundamental failings of conventional thinking about software requirements and some suggestions for getting it right / T. Glib // J. Software Engineering & Applications. - 2010. – № 6. – pp. 827-838.
2. Object Management Group, UML Resource Page (электронная ссылка) // www.uml.org – 2008.
3. Gruber, T.R. A translation to portable ontologies [текст] // Knowledge Acquisition. – 1993. – № 5(2). – С. 199-220.
4. IEEE 830-1998, «IEEE Recommended Practice for Software Requirements Specifications» (электронная ссылка) // www.standarts.ieee.org/findstds/standard/830-1998.html.