

АХМАД АЛИ (АБДЕЛЬ КАРИМ) АЛЬМХЕРАТ, аспирант,
Г. Н. ЖОЛТКЕВИЧ, д-р техн. наук, ХНУ имени В. Н. Каразина
А. Г. ЖОЛТКЕВИЧ, ООО «Совместное украинско-словенское
предприятие «МОНИС»

МОДЕЛИРОВАНИЕ ОБМЕНА ИНФОРМАЦИЕЙ В ИНФОРМАЦИОННЫХ СИСТЕМАХ: КОНЦЕПТУАЛЬНЫЙ УРОВЕНЬ

У статті розглянута проблема імітаційного моделювання процесів обміну інформацією у розподілених програмних комплексах, зокрема інформаційних системах. Обґрунтовано необхідність створення інструментарію для забезпечення такого моделювання з метою визначення адекватності того чи іншого архітектурного шаблону до конкретної проектної ситуації. Запропонована концептуальна модель предметної області імітаційного моделювання процесів обміну інформацією в інформаційних системах.

1. Постановка общей проблемы

Растущая сложность задач, для решения которых применяются информационно-вычислительные системы, приводит к необходимости интеграции программных комплексов в единую систему. Возникающие в результате интеграции системы являются, как правило, системами с распределенными данными и их обработкой. Существует ряд архитектурных шаблонов проектирования таких систем [1], однако обоснование их применения носит умозрительный, а не технический характер. В связи с этим возникает проблема оценки эффективности использования того или иного архитектурного шаблона на фазе проектирования системы.

2. Анализ публикаций, постановка задачи

Одной из основных концепций программной инженерии является концепция повторного использования решений на разных уровнях: код, структура, проектное решение, архитектура [2]. Использование шаблонов (паттернов) проектирования при разработке программного обеспечения с момента появления работы [3] стало одним из основных инструментов повышения эффективности разработки за счет повторного использования на уровне архитектурных и проектных решений. Роль шаблонов проектирования как современного средства разработки программного обеспечения отмечена также в [4]. Детально шаблоны проектирования архитектурного уровня для корпоративных программных систем описаны в [5]. В работе [6] предложено архитектурное решение, ориентированное на интеграцию информационных систем на уровне корпорации с сохранением суверенитета каждого из интегрируемых компонентов над своим информационным ресурсом. Общей чертой всех указанных работ является подробное описание различных

архитектурных решений при отсутствии рекомендаций относительно методов оценки этих решений с целью обоснованного выбора какого-либо из них.

В связи с вышеизложенным, актуальной является задача разработки методологии объективной оценки возможностей и последствий использования существующих шаблонных решений для конкретной проектной ситуации. В настоящей работе предлагается решить эту задачу путем имитационного моделирования поведения распределенной информационной системы. Авторы предлагают описание подхода к имитационному моделированию информационных систем на концептуальном уровне. В работе также вводятся основные характеристики, анализ которых по результатам моделирования позволяет сравнить возможные последствия использования различных архитектурных шаблонов.

3. Основная часть

Несмотря на разнообразие развитых систем имитационного моделирования как общего назначения (например, GPSS), так и ориентированных на конкретные приложения (например, система NetCracker ориентирована на моделирование процессов в вычислительных сетях и сетях связи) [7], авторам не известны имитационные системы, язык которых был бы адекватен задачам моделирования процессов обмена данными в информационных системах.

Прежде всего, следует подчеркнуть, что мы будем рассматривать информационную систему как множество взаимодействующих элементов. Эти элементы мы будем называть капсулами (*Capsule*). Капсула представляет собой сложный, возможно распределенный, объект, который взаимодействует со своим окружением [4]. Капсула, с нашей точки зрения, может иметь закрытый для внешнего доступа информационный ресурс (в работе [6] рассматривается аналогичный объект, названный «информационным объектом»), структурно разделенный на компоненты данных: собственные данные (класс **OwnDataUnit**) и данные, являющиеся собственностью других капсул (коллекция объектов класса **AdoptedDataUnit**), используемые рассматриваемой капсулой (см. рис. 1).

При использовании, например, архитектурного шаблона клиент/сервер, очевидно, что клиентские капсулы не имеют собственного информационного ресурса, серверная же капсула монополюно владеет всеми информационными ресурсами системы.

Капсула является не просто хранилищем, но и совокупностью операторов на данных, обладая способностью выполнять задачи (*Task*) по их обработке (см. рис. 1: метод `perform()` класса *Task*). Эти задачи целесообразно разбить на два класса: транзакции (**Transaction**) – задачи, обеспечивающие модификацию данных информационного ресурса, и запросы (**Query**) – задачи, обеспечивающие доступ к данным для чтения. Следует отметить, что наличие у капсулы копий «чужих» данных приводит к необходимости синхронизации таких копий с оригиналом. Это обосновывает введение

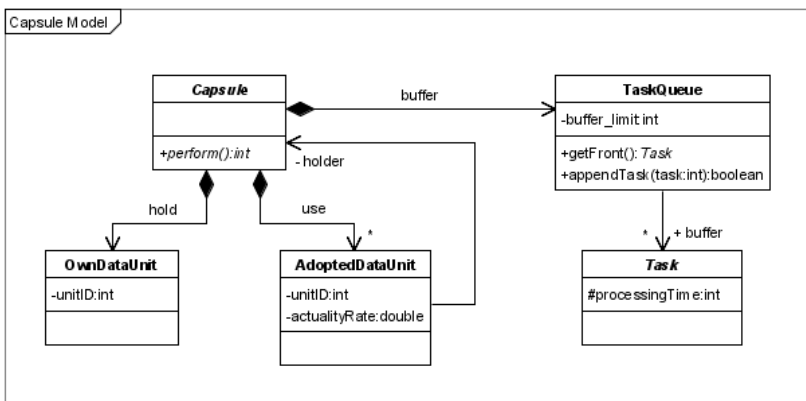


Рис. 1. Концептуальная модель

специального класса задач – синхронизаторов (**Synchro**), которые запрашивают актуальное состояние компонента данных у его владельца и, получив его, осуществляют обновление этого компонента. Мы предполагаем, что использование синхронизатора является единственным способом модификации не собственных компонентов данных капсулы.

В процессе автономного функционирования капсулы происходит постепенное снижение уровня актуальности данных «чужих» данных. Для описания текущего уровня актуальности объекта класса **AdoptedDataUnit** используется атрибут *actualityRate*, значение которого интерпретируется как вероятность получения актуального отчета на основании запроса к рассматриваемому объекту.

Полученные капсулой для исполнения задачи помещаются в очередь задач капсулы (класс **TaskQueue**), и упомянутый выше метод *perform()* при освобождении исполнительного механизма капсулы снимают с головы очереди задач капсулы задачу для обработки. При этом капсула переходит на время, определяемое атрибутом *processingTime* класса наследника класса **Task**, в состояние занятости (см. рис. 2).

Предполагается, что помещение новой задачи в очередь задач капсулы происходит только в том случае, если текущий объем буфера не превосходит значения атрибута *buffer_limit* класса **TaskQueue**. Следует обратить внимание, что для определенных задач моделирование этот атрибут может принимать значение *+infinity*.

Концептуальная модель задачи приведена на рис. 2.

Комментариев требует, на наш взгляд, только структура класса **Synchro**. В структуре этого класса инкапсулированы два объекта классов **Query** и **Transaction** соответственно.

Сформулируем теперь основные задачи, которые можно решать путем постановки имитационных экспериментов на моделях, построенных в рамках

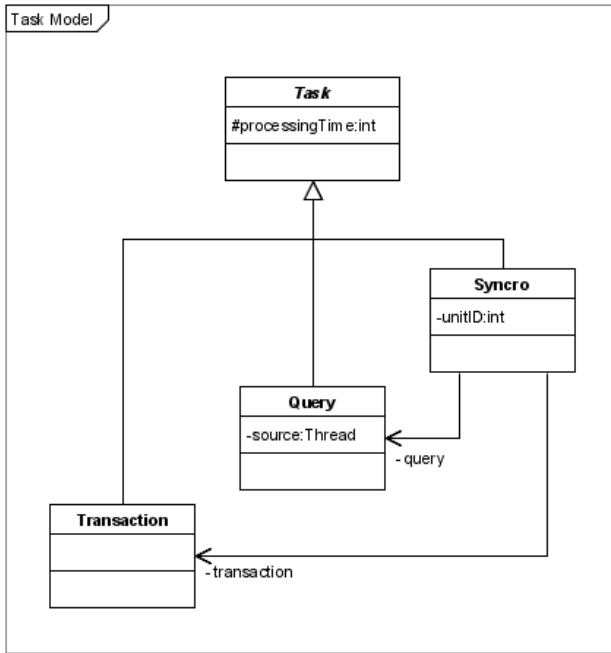


Рис. 2. Концептуальная модель

каркаса имитационного моделирования, базирующегося на решениях, описанных выше.

Во-первых, метод `perform()` класса *Capsule*, который в силу описанной сигнатуры (см. рис. 1) при реализации должен возвращать целое число – длительность в единицах модельного времени – обеспечивает решения задачи, которая поступила для выполнения, позволяет измерять временные характеристики системы.

Во-вторых, наличие атрибута `buffer_limit` для класса *TaskQueue* обеспечивает вычисление вероятности потери задачи в системе.

В-третьих, наличие атрибута `actualityRate` для класса *AdoptedDataUnit* позволяет решать задачу оценки вероятности принятия неправильного решения системой за счет использования устаревших данных.

Таким образом, основными измеряемыми в процессе имитационного эксперимента характеристиками системы в рамках предложенной концептуальной модели системы имитационного моделирования являются: риск принятия неправильного решения в результате выполнения той или иной задачи, который определяется вероятностью принятия неправильного решения за счет десинхронизации данных; риск потери задачи в очереди ожидания выполнения, который определяется вероятностью такого события; среднее время ожидания выполнения процесса.

4. Выводы

Таким образом, в работе предложена концептуальная модель каркаса имитационного моделирования поведения информационных систем с целью определения целесообразности использования тех или иных архитектурных шаблонов при их проектировании.

Концептуальная модель базируется на следующих предположениях:

1. Структурно-логически архитектура информационной системы может быть представлена множеством взаимодействующих капсул.
2. Каждая капсула инкапсулирует данные двух типов – данные, которыми она владеет, и данные которыми она пользуется.
3. Данные, которыми капсула владеет, могут свободно ее модифицироваться (т. е. над ними возможно выполнение транзакций в рамках этой капсулы), в то время как данные, которыми капсула пользуется, доступны только для чтения (т. е. над ними возможно выполнение только запросов в рамках этой капсулы).
4. Исключение при работе с данными, которыми капсула пользуется, возникает при работе с синхронизаторами. Синхронизаторы инкапсулируют как запрос, так и транзакцию, и только такая транзакция может быть выполнена над данными, которыми капсула пользуется, в рамках этой капсулы.
5. Эффективность исследуемой архитектуры определяется путем наблюдения в рамках имитационного эксперимента следующих характеристик – риск принятия неправильного решения в результате выполнения той или иной задачи, который определяется вероятностью принятия неправильного решения за счет десинхронизации данных; риск потери задачи в очереди ожидания выполнения, который определяется вероятностью такого события; среднее время ожидания выполнения процесса.

Дальнейшей своей задачей авторы видят разработку динамических моделей каркаса имитационного моделирования поведения информационных систем.

Список литературы: 1. Таненбаум Э., ван Стен М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877 с. 2. Соммервилл И. Инженерия программного обеспечения. М.: Вильямс, 2002. – 624 с. 3. Гемма Э., Хелм Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2003. – 366 с. 4. Грэхем И. Объектно-ориентированные методы. Принципы и практика. Третье изд. – М.: Вильямс, 2004. – 879 с. 5. Фаулер М. Архитектура корпоративных программных приложений. – М.: Вильямс, 2004. – 544 с. 6. Альмхерат Ахмад Али (Абдель Карим), Жолткевич Г. Н., Игнатов С.Ю. Об одном подходе к интеграции неоднородных информационных ресурсов // Вісник Харк. нац. ун-та. – № 629, 2004: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління. – Вип. 3. – С. 48 – 55. 7. Рыжков Ю. Н. Имитационное моделирование. Теория и технологии. – СПб.: Корона Принт, 2004. – 380 с.

Поступила в редколлегию 03.04.06