УДК 614.2+574/578+004.38

А.А. Литвинов, Д.Л. Грузин, П.П. Гуреев

ОСОБЕННОСТИ АВТОМАТИЗАЦИИ РАЗРАБОТКИ ФУНКЦИЙ В МНОГОУРОВНЕВЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

Аннотация. В работе предлагается идея подхода направленного на упрощающее создание функциональных компонентов, отвечающих за полную реализацию функции. После описания разрабатываемой функции в упрощенной форме, программист получает перечень и порядок работ необходимых для завершения выполнения функции с необходимой инфраструктурой: готовыми и/или полу-готовыми программными компонентами.

Актуальность темы. На текущий момент важной практической задачей разработчиков ПО является быстрое создание и сопровождение качественного многоуровневого программного обеспечения, как правило, представляющего вариант распределенной информационной системы. Сервисная концепция предполагает, что разработанный продукт будет отвечать характеристикам качества, таким как: устойчивость, тестируемость, полезность, доступность, масштабируемость, открытость, гибкость. Это накладывает на процесс разработки дополнительные ограничения/правила: следование шаблонам и стилям; документирование разработки на различных уровнях;покрытие разрабатываемых компонентов, модулей, подсистем различного вида тестами; управление проектами, процессами. Построение современных систем, как правило, требует участия большого кол-ва различных специалистов и хорошо организованного процесса, гарантирующего качество разработки [1]. Важнейшей составляющей такого процесса является анализ и использование опыта предыдущих разработок, включающего аспекты планирования, управления, создания и реализацию технических решений.

В целом процесс разработки информационной системы сводится к трансформации функциональных требований пользователя и искомых атрибутов качества системы в готовое решение: множество

 ISSN 1562-9945

взаимодействующих программных компонентов, работающих в операционном окружении. В основе данной трансформации лежит процесс, включающий множество активностей, полностью или частично формализованных и покрывающих вопросы генерации, выполнения и контроля за выполнением задач. Генерация множества задач диктуется архитектурой системы [2] и базируется на типовых решениях, составляющих технологическую основу уровня зрелости компании. Проверка соответствия системы требованиям базируются наформализованных тестах приемки (useracceptancetests), которые могут быть автоматизированы. Как правило, каждая из задач трансформируется в набор модульных или интеграционных тестов, которые служат для:проверки компонента на предмет выполнения контракта, документирования возможностейкомпонента.

Первая задача автоматизации построения решений состоит в том, чтобы для поставленной задачи сформировать набор компонентов, включая обязательные составляющие (интерфейсы, тесты, реализацию, исключения). Вторая задача – преобразование функции, как правило уровня c-requirement, в набор задач, согласно выбранной архитектуре: в соответствии одной из веток сценария, описанного на языке L, используя предыдущий накопленный опыт, формализованный в виде трансформаций «L-T», система должна предложить полный или частичный набор компонентов.И в первом и во втором случае важным вопросом является обучение системы проводить такие преобразования, при этом средство должно быть доступным и удобным для разработчика. В данной работе рассматривается вариант решения первой задачи.

Анализ последних публикаций. В работе [3] была представлена интерпретация модельно-ориентированного подхода, упрощающий создание и сопровождение информационной системы, основой которого является модель, описанная с использованием фреймового подхода представления знаний. На базе описанной модели производится генерация ряда компонентов, что существенно упрощает процесс разработки многоуровневых приложений, а также производится интерпретация модели в ходе работы системы, что исключает необходимость построения и тестирования дополнительных классов. В работе [4] рассмотрены вопросы формирования функциональных компонентов, в

основе которой лежит использование шаблонов, обеспечивающих генерацию полноценных функциональных компонентов.

Постановка задачи. В данной работе предлагается вариант автоматизации трансформации функции в набор компонентов, отвечающих за ее реализацию.

Основная часть. Реализация функции базируется на выделении задач, множество которых диктуется архитектурой системы (уровнями и типами компонентов-классов, используемых в рамках данных уровней). Компонент представляет собой следующий набор: контракт, реализация, модульные тесты, при этом контракт включает три составляющих: интерфейс, исключения, домен, амодульные тесты направлены на тестирование контракта.Важной особенностью, является то, что компоненты распределены по классам, составляющим шаблоны решения задач уровня разработчика (Helper, Manager, Strategy, Validator, Sender, Receiver, Processor и т.п.).

Неотъемлемой частью всех подходов, направленных на совершенствование качества процесса разработки, является выделение и анализ лучших практик после завершения стадии или реализации продукта, с целью выделения стандартов решения задач, составляющих базу типовых решений. Создание такой базы, не только повышает уровень повторного использования компонентов, повышает скорость разработки, но и создает основу для автоматизации, что переводит процесс разработки на новый качественный уровень.

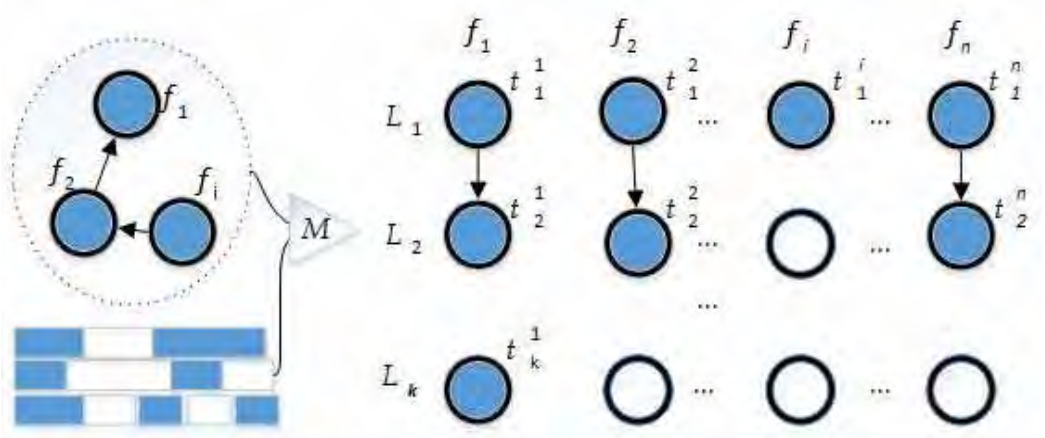

Рисунок 1 - Отображение функций компонентом: *f*- функция, *M* – механизм отображения функций в решение, *t* – задачи, *L* – уровень системы

 ISSN 1562-9945

Предлагаемая идея базируется настандартизации(типизации) функций по критериям их связи с уровнями системы: определении наборов типовых задач, связанных с реализацией функции и формировании множества шаблонов. Такой шаблон можно определить, как средство описания отображения исходного (доменно-ориентированного) описания, заданного с использованием языка, включающего в себя целевой и множество семантических маркеров, описывающих преобразование, в группу конструкций близких к исполняемой форме (файлов, классов, проектов, скриптов создания и изменения структуры базы данных). Детали применения шаблонов описаны в работах [4].

$$m_j : f_i \to \tau_i, \quad \tau_i \subseteq T \subseteq F \times L, \; f_i \in F, \; m_j \in M \; , \tag{1}$$

$$t_i \in F \times L, \; f_j \in F \; . \tag{2}$$

Следует отметить, что центральную роль в этом процессе играют структуры данных: хранимые сущности, доменные объекты, различного рода представления, например, DTO-объекты. Большинство функций связано с одним или несколькими доменными объектами, которые составляют основу для структур другого рода.

Формирование конфигурации, описывающей решение, базируется на задании набора уровней, задействованных в реализации функции и роли компонентов в рамках данных уровней. Компоненты могут быть как новые, так и уже существующиеи требующие внесения изменений.Важной составляющей здесь является не только генерация инфраструктуры (компонентов, интерфейсов, тестов), но и планирование работ, необходимых для реализации функции.

Действия разработчика представляется следующим образом:задание доменных объектов, выбор типа функции с указанием минимума конфигурационной информации (включение или исключение необходимых компонентов, используемых уровней либо просто указание набора уровней), запуск трансформации. В результате получается набор компонентов, разной степени полноты завершенности, отвечающих за решение задач, выполнение которых диктуется выбранной функцией.

Рассмотрим на примере простейшую задачу получениясписка пользователей для серверной части системы, состоящей из слоев рисунке 2.Скрипт, описывающийзаданиеимеетвид:

F:GetUserCollection D:UserBase DTO:UserInfo SL:[h.Main, controller.UserInfo, bl.User, da.User] Out:collectionIn:void

На базе данного скрипта, с использованием уже готовых шаблонов будут созданы необходимые методы в заданных классах (табл. 1), с соответствующей логикой тестирования логики. После получения такой поддержки, разработчику остается довести полученную инфраструктуру до рабочего состояния, при этом нет необходимости что-либо вспоминать или отвлекается, что снижает сложность выполняемых задач. Для более сложных задач кроме кода генерируются пошаговые рекомендации по выполнению работ.

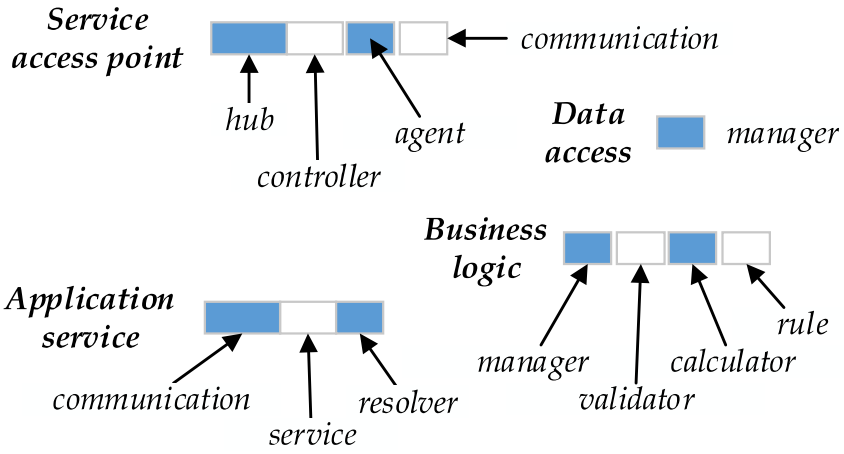

Рисунок 2 - Структура основных уровней серверного приложения

Таблица 1

Порожденные методы

Структура	Метод
MainHub	Task<List<UserInfo>>GetUserCollection()
UserInfoController	List<UserInfo>GetUserInfoCollection()
UserManager	List<UserBase>GetUserCollection()
UserDalManager	List<UserBase>GetCollection()
UserTransformer	UserInfoTransform(UserBaseuser)

Вывод. Предложенный подход упрощает планирование и реализацию функций в системах с многоуровневой архитектурой. Подход базируется на решениях, представленных в работах [3, 4]. После описания разрабатываемой функции, путем задания структур данных и уровней участвующих в ее реализации, с указанием дополнительных аспектов, связанных с компонентами в рамках данных уровней, программист получает перечень и порядок работ, необходимых для завершения выполнения функции (возможна также оценка времени и

 ISSN 1562-9945

усилий для ее успешного завершения) с необходимой инфраструктурой: готовыми и/или полу-готовыми программными компонентами. Это создает основу разработки, уменьшает количество ошибок, помогает планированию завершения задач при построении сложных многоуровневых системах. Предложенный подход наиболее эффективен в случае предварительного детального проектирования компонентов, наполненности базы решений-шаблонов.

ЛИТЕРАТУРА

1. Mary Beth Chrissis. CMMI® for Development Guidelines for Process Integration and Product Improvement, Addison-Wesley Professional; 3 edition. – 688 p.
2. ANSI/IEEE 1471-2000, “Recommended Practice for Architecture Description of SoftwareIntensive Systems.”
3. Литвинов О.А. Оптимизация процесса разработки многоуровневых программных компонентов /Литвинов О.А., Грузин Д.Л., Вякилов А.С.// Системные технологии. Региональный межвузовский сборник научных работ. –2014. – Выпуск 1(90). – С. 29-35.
4. Литвинов О.А. Особливості автоматизації процесу розробки функціональних компонентів інформаційної системи / Литвинов О.А., Грузін Д.Л. // Системні технології. Регіональний міжвузівський збірник наукових праць. –2015.– Випуск 1(96). –С. 78-86.