УДК 004.415.23

АРХИТЕКТУРНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СИСТЕМ СТРУКТУРНЫМ МЕТОДОМ

С.Ц. Манжикова

Рассматривается специальный прием преобразования DFD модели проектируемой программной системы, который позволяет сделать логически обоснованный переход к этапу архитектурного проектирования ПО. Анализ нижнего уровня декомпозиции DFD модели приводит к выявлению так называемых центров преобразования (транзакций) информационных потоков в потоки воздействий ПС на входные потоки данных. Каждый центр транзакции образует в программной системе свою «область влияния», которая может быть реализована в виде конкретного программного компонента. Таким образом, DFD модель становится основой соответствующего архитектурного решения программной системы. При этом обеспечивается корректность архитектурного решения ПО, удобство его эксплуатации и модификации при сопровождении. Последнее является едва ли не самым важным требованием к современным сложным ПС. Приводится пример итеративного построения архитектуры программной системы (ПС) на основе анализа диаграмм потоков данных DFD. Показано, что программные компоненты на первой итерации процесса разработки архитектуры ПО могут наследоваться от процессов/функций DFD модели. Сформулированы правила преобразования DFD для определения типа архитектуры, наиболее отвечающего проектируемому ПО.

Ключевые слова: проектирование ПО; структурный метод; архитектура программы; анализ DFD; центр преобразования потоков данных.

ПРОГРАММАЛЫК СИСТЕМАЛАРДЫ ТҮЗҮМДҮК МЕТОД МЕНЕН АРХИТЕКТУРАЛЫК ДОЛБООРЛОО

С.Ц. Манжикова

Макалада долбоорлонгон программалык системанын DFD моделин кайра түзүүнүн атайын ыкмасы каралат, ал программалык камсыздоону архитектуралык долбоорлоо баскычына логикалык негиздүү өтүүгө мүмкүндүк берет. DFD моделинин декомпозициясынын төмөнкү деңгээлине талдоо жүргүзүү маалымат агымдарынын кайра түзүү борборлорун (транзакцияларын) аныктоого алып келет. Транзакциянын ар бир борбору программалык системада өзүнүн «таасир этүү чөйрөсүнө» ээ, ал конкреттүү программалык компонент түрүндө ишке ашырылышы мүмкүн. Ошентип, DFD модели программалык системанын тиешелүү архитектуралык чечилишинин негизи болуп калат. Ошол эле учурда программалык камсыздоонун архитектуралык чечилишинин тууралыгы, техникалык тейлөө учурунда аны эксплуатациялоонун жана модификациялоонун ыңгайлуулугу камсыздалат. Акыркысы заманбап татаал программалык камсыздоолорго карата эң маанилүү талаптардын бири болуп эсептелет. DFD маалымат агымынын диаграммаларын талдоонун негизинде программалык системанын архитектурасынын (PS) кайталанып курулушунун мисалы келтирилген. Программалык камсыздоонун архитектурасын иштеп чыгуу процессинин биринчи кайталануусундагы программалык компоненттер DFD моделинин процесстеринен / функцияларынан мурас болуп өткөндүгү көрсөтүлгөн. DFDди трансформациялоо эрежелери иштелип чыккан программалык камсыздоого дал келген архитектуранын түрүн аныктоо үчүн иштелип чыккан.

Түйүндүү сөздөр: программалык камсыздоону долбоорлоо; түзүмдүк метод; программанын архитектурасы; DFD талдоо; маалымат агымын трансформациялоо борбору.

ARCHITECTURAL DESIGN OF SOFTWARE SYSTEMS VIA STRUCTURAL METHOD

S.Ts. Manzhikova

The article discusses a special technique for transforming the DFD model of the designed software system, which allows you to make a logical transition to the stage of software architectural design. The analysis of the lower level of the DFD model decomposition leads to the identification of the so-called centers of transformation (transactions) of

information flows into flows of software system effects on input data flows. Each transaction center forms its own “area of influence” in the software system, which can be implemented as a specific software component. Thus, the DFD model becomes the basis for the corresponding architectural solution for the software system. This ensures the correctness of the architectural solution for the software, the ease of its functioning and modification during maintenance. The latter is perhaps the most important requirement for modern complex software systems. An example of iterative construction of the architecture of a software system based on the analysis of DFD data flow diagrams is given. It is shown that software components at the first iteration of the software architecture development process can inherit from the processes / functions of the DFD model. DFD transformation rules are formulated to determine the type of architecture that best suits the designed software.

Keywords: software design; structural method; software architecture; DFD analysis; data flow transformation center.

Структурный метод проектирования/разработки программных систем (ПС) является первым исторически сложившимся и научно обоснованным подходом в современной программной инженерии [1]. Более того, многолетний опыт преподавательской работы показывает, что изучение и освоение этого метода логически должно предшествовать изучению и освоению более позднего объектно-ориентированного анализа и проектирования ПС (ООА и П ПС), потому что идеи исследования и моделирования сложных систем почти осязаемо реализованы в программных продуктах для их автоматизированного анализа и разработки – в программных пакетах All Fusion Business Process Modeler, ERwin Data Modeler, Oracle Designer и др. Однако студенты почти всегда испытывают существенные затруднения при решении задачи декомпозиции сложных систем, в том числе и ПС при их разработке с применением инструмента/стандарта DFD (Data Flow Diagram – диаграмм потоков данных). То, что основной/главный процесс/функция должен быть декомпозирован на более простые процессы, или подпроцессы/подфункции – это понятно, и чаще всего задача определения структуры 1-го уровня детализации/декомпозиции DFD студентами успешно решается. Но какие из подпроцессов 1-го уровня детализации следует продолжить декомпозировать, как определить эти процессы? Насколько это необходимо в практике программной инженерии?

В данной работе рассматривается пример проектирования программного обеспечения (ПО), формулируются рекомендации по решению поставленных вопросов и иллюстрируется логическая связь между структурой DFD и архитектурным решением ПС.

В работе [2] для иллюстрации структурного метода разработки приводится пример «Банковской задачи». На 1-ом уровне декомпозиции показаны четыре подпроцесса. Второй же уровень детализации разработан только для одного из них. Почему не для всех? Какими особенностями обладает подпроцесс, для которого производится дальнейшая детализация?

В работе [3] предложены «правила» распознавания таких процессов. Но чтобы применение таких правил стало обоснованным, необходимо по-другому изобразить детализированную DFD, а именно, подпроцессы на диаграмме следует преимущественно расположить по диагонали слева сверху направо вниз так, чтобы для каждого подпроцесса четко выделялись входные и выходные потоки данных/информации. Для банковской задачи DFD 1-ого уровня детализации будет иметь вид, показанный на рисунке 1.

Здесь сохранена индексация подпроцессов. Для наглядности сохранены изображения внешних сущностей *Клиент* и *Компьютер банка*. Представлены все потоки информации, документов и денег.

Рассмотрим, как преобразуется входной (входные) поток в выходной (выходные) поток. Простой входной поток *Кредитная_карта* на выходе подпроцесса 1.4 остается элементарным, для которого можно определить тип данных, например, *integer*. Входные потоки для подпроцесса 1.1 преобразуются им также в некий простой поток *Запрос_на_обслуживание*. Входные потоки процессом 1.2 преобразуются тоже в элементарный поток *Денежная_сумма.* А вот входные потоки для подпроцесса 1.3 трансформируются не в один, а в три выходных потока, каждый из которых является путем/каналом действия/воздействия системы ПО. Такой процесс является ***центром транзакции (преобразования) и должен детализироваться.*** Поэтому для него в работе [2] приводится 2-ой уровень детализации. В нашем случае декомпозиция подпроцесса 1.3 представлена на рисунке 2.

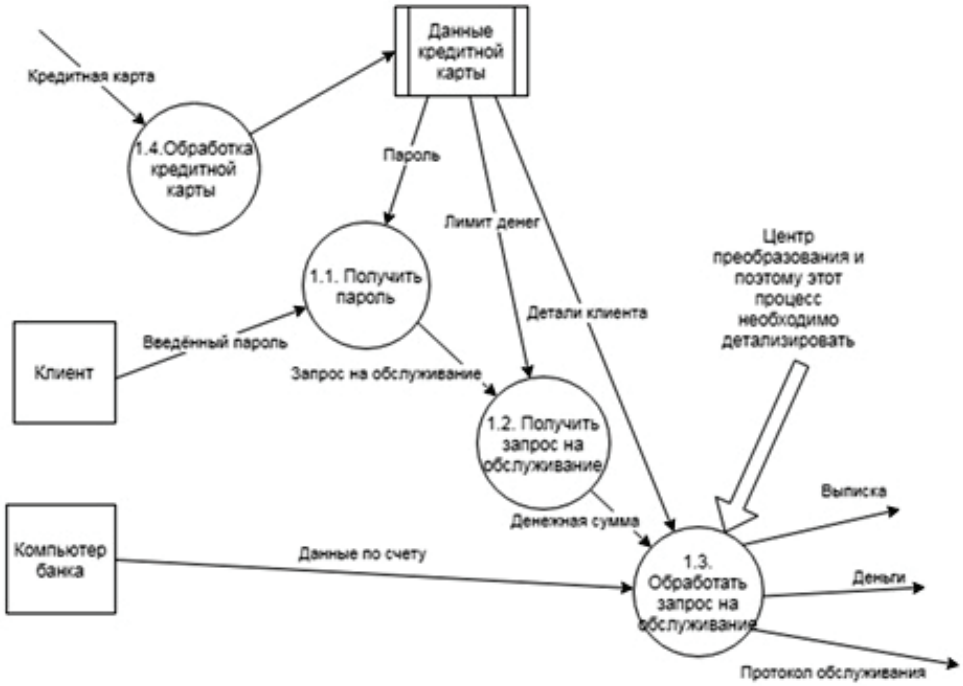

Рисунок 1 – Первый уровень детализации «Банковской задачи»

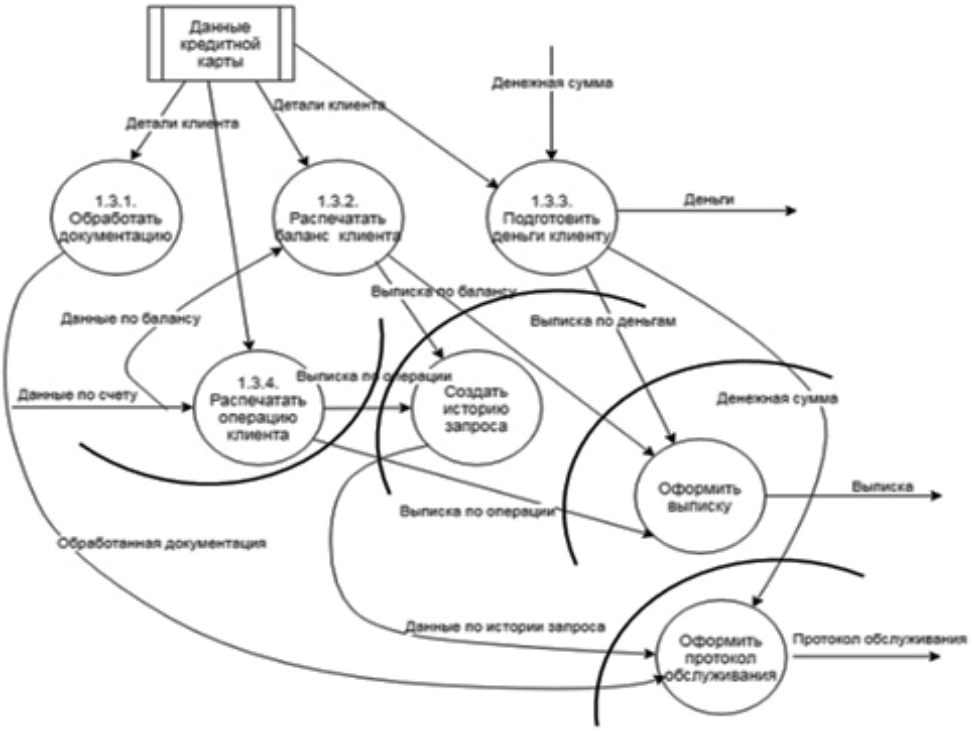

Рисунок 2 – Второй уровень детализации «Банковской задачи» – детализация подпроцесса 1.3

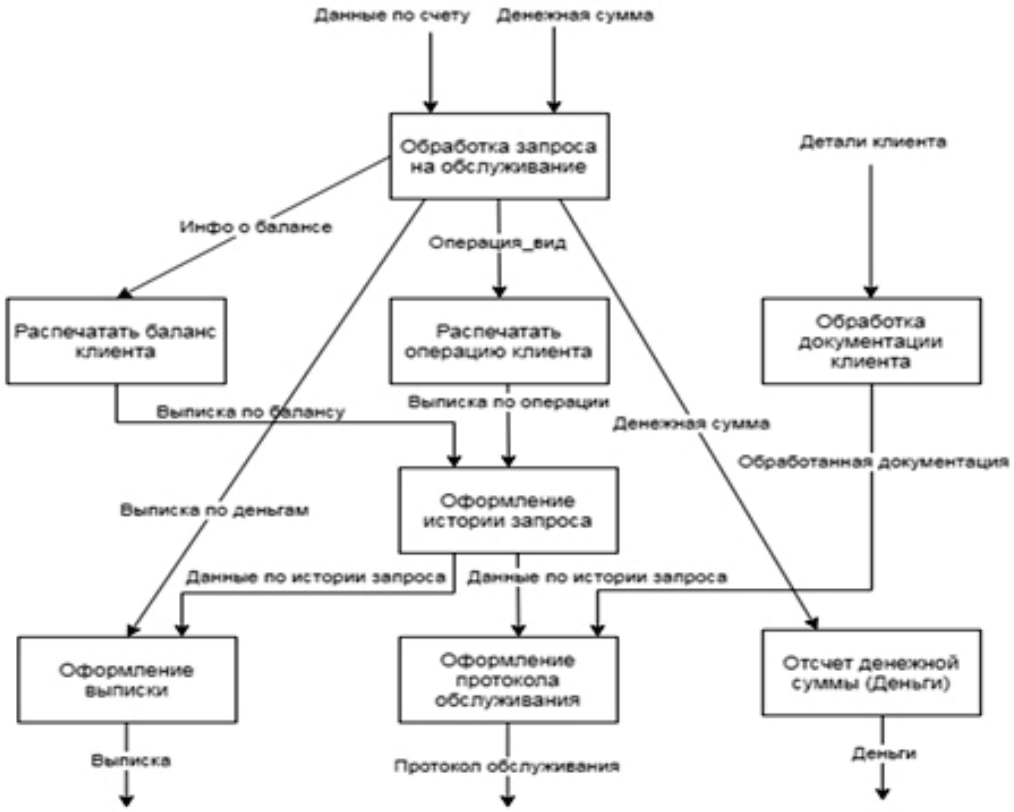

Рисунок 3 – Первая итерация структуры приложения «Банковской задачи»

Все подпроцессы 2-го уровня декомпозиции являются понятными для того, чтобы реализовать их в виде функционала ПО, поэтому дальнейшая их детализация не целесообразна – это правило остановки операции декомпозиции.

Поскольку каждый подпроцесс можно рассматривать как центр преобразования входного потока в выходной внутри ПС, то можно выделить «зону его влияния». Это показано дугами. В результате образуется ***карта преобразований***, которая является основой для построения первой итерации модульной структуры ПО, показанной на рисунке 3.

В соответствии с картой (рисунок 2) определены компоненты/модули ПО нижнего уровня: *Оформление выписки, Оформление протокола обслуживания и Отсчет денежной суммы* (рисунок 3). Компонентами более высоких уровней становятся модули (рисунок 3). Оформление *истории запроса; Распечатать баланс клиента; Распечатать операцию клиента; Обработка документации клиента и Обработка запроса на обслуживание*, соответствующие процессам с аналогичными именами (рисунок 2).

Направление потоков данных, а значит и управлений в ПС, свидетельствует о ***проектировании ПС сверху вниз.*** Такой подход повышает корректность самого процесса разработки ПО и обеспечивает его модифицируемость, что имеет очень большое значение для качественного сопровождения ПО в последующем. Анализ первой итерации структуры приложения «Банковская задача» (рисунок 3) позволяет выделить в ней слои, характерные для классической трехслойной архитектуры (рисунок 4).

Это слой ***Уровня доступа к источникам данных,*** слой ***Уровня бизнес логики*** и слой ***Уровня доступа к приложению*** (пользовательский уровень).

Необходимо особо подчеркнуть, что ***полученная архитектура корректна и удобна для успешной реализации и эксплуатации ПС, а также возможной её модификации в процессе сопровождения.***

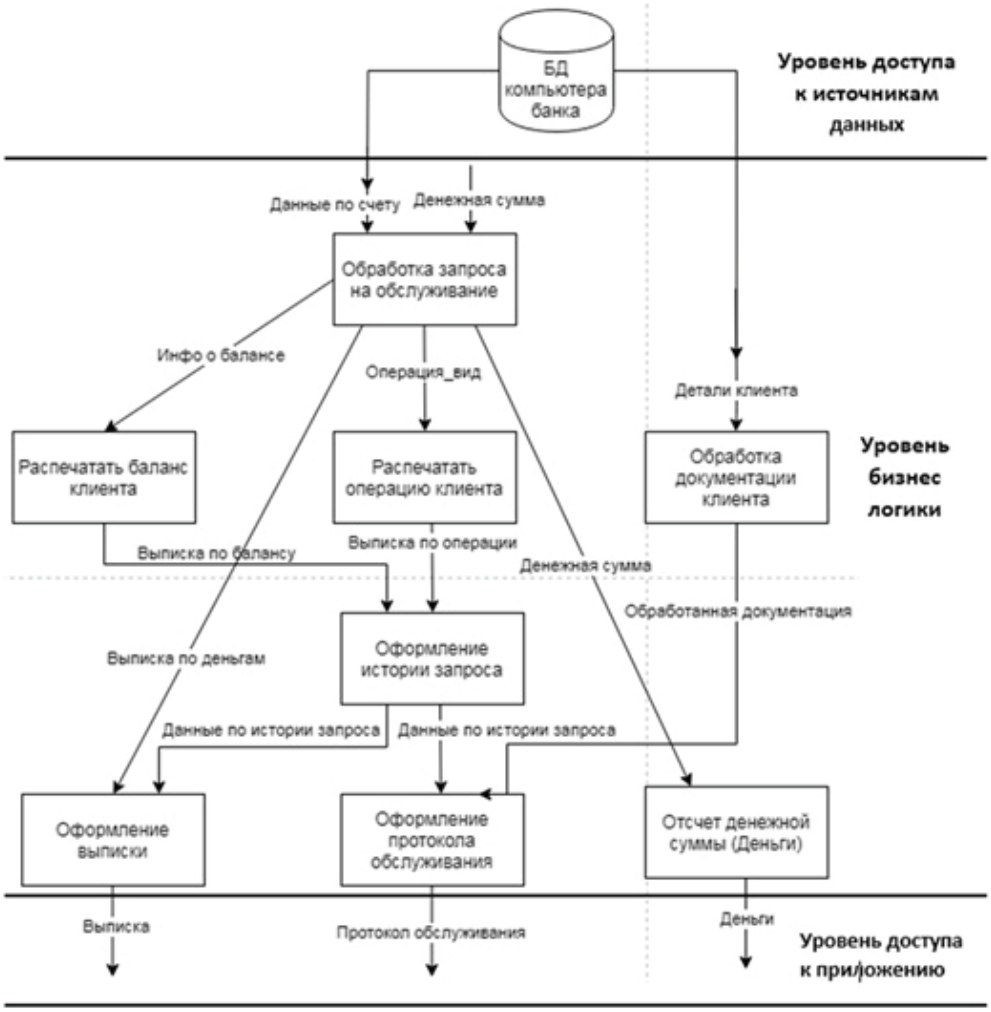

Рисунок 4 – Архитектура приложения «Банковская задача»

Это явилось результатом применения для разработки ПО структурного подхода и DFD как конкретного инструмента этого метода.

Включение в программу учебного процесса приема предварительного проектирования архитектуры ПС на основе анализа и преобразования соответствующей DFD модели, изложенного выше, будет способствовать реализации требуемых [4] профессиональных и общенаучных компетенций специалистов в области программной инженерии.

Литература

1. *Орлов С.А.* Программная инженерия / С.А. Орлов. СПб.: Питер, 2018. 640 с. (Серия «Учебник для вузов»).
2. *Калянов Г.Н.* CASE-технологии: консалтинг в автоматизации бизнес-процессов / Г.Н. Калянов. М.: Горячая линия, 2003. 320 с.
3. URL: https://ebookpdf.com/roger-s-pressman Roger S. Pressman. Software Engineering: A Practitioner's Approach. 7-th Edition, 2020.
4. ФГОС ВО (3++) по направлению подготовки 09.04.04 Программная инженерия. URL: https://regulations.tusur.ru/documents/934