УДК 681.3.01
DOI 10.26732/2618-7957-2018-2-100-104

АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ВЫСОКОНАДЕЖНЫХ СИСТЕМ

А. В. Караванов, Н. Д. Иванов
Сибирский федеральный университет,
г. Красноярск, Российская Федерация

В статье рассматривается проблема построения программного обеспечения для высоконадежных систем. Исследуются основные критерии, позволяющие оценить эффективность спроектированной архитектуры программного обеспечения. Приводится их краткое описание. Программное обеспечение с большим спектром выполняемых задач разрабатывается по модульному типу. Программа делится на модули по их функциональному назначению. Предлагается выделить модуль в отдельную программу для повышения отказоустойчивости программного обеспечения в высоконадежных системах. Приводится пример построения предложенной архитектуры с разбиением программного обеспечения на модули (программы) по функциональному назначению. С этой целью разработана программа-посредник, при помощи которой модули обмениваются информацией. Взаимодействие между модулями и программой-посредником может происходить при помощи различных технологий (TCP/IP, общие файлы, разделяемая память и т. д.). В приведенном примере взаимодействие было реализовано при помощи протокола сетевого взаимодействия TCP/IP, а также проведен эксперимент для сравнения монолитно построенной программы и программы, сделанной по предложенной архитектуре. В рамках эксперимента источник данных передавал пакеты через программное обеспечение, реализованное на основании предложенной архитектуры. За критерий надежности программного обеспечения принято количество доставленных пакетов. Данный эксперимент подтвердил преимущество надежности программного обеспечения, построенного по предложенной архитектуре.

Ключевые слова: архитектура программного обеспечения, программный модуль, декомпозиция программ, высоконадежные системы.

Введение

Разработка программ – довольно сложный и трудоемкий процесс, в котором проектирование корректной и надежной архитектуры (структуры) играет ключевую роль. Распределение и координация усилий по созданию программ в группе разработчиков часто оказываются наиболее ответственными и трудными решениями, т. к. влияют на основной результат. Целью проектирования архитектуры является определение внутренних свойств программного обеспечения (ПО) и детализация его внешних свойств на основе выданных заказчиком и впоследствии проанализированных требований.

Правильно выбранная архитектура ПО помогает сделать процесс разработки и сопровождения программы более простым и эффективным. К критериям хорошо спроектированной архитектуры ПО относят:

• *Эффективность программы*. В первую очередь программа должна решать поставленные задачи и хорошо выполнять свои функции, причем в различных условиях. Сюда можно отнести такие характеристики, как надежность, безопасность, производительность, способность справляться с увеличением нагрузки (масштабируемость) и т. п. [1];

• *Изменяемость программы*. Любая программа подвергается изменениям с течением временем. Чем быстрее и удобнее можно изменить программу, тем она гибче и конкурентоспособнее [2]. Изменение одного модуля программы не должно влиять на другие модули;

•*Расширяемость программы*. Возможность добавлять в программу новый функционал, не нарушая ее основной структуры. На первом этапе проектирования в программу необходимо добавлять только самый необходимый функционал, но при этом архитектура должна позволять легко добавлять новый функционал по мере необходимости [3]. Программу следует проектировать так,

чтобы изменение и добавление новых функций достигалось бы за счет написания нового кода (расширения) и при этом не приходилось бы менять уже существующий;

• *Масштабируемость процесса разработки*. Возможность уменьшить срок разработки за счёт привлечения новых людей. Архитектура должна позволять распараллеливать процесс разработки так, чтобы множество людей могли работать над ним одновременно [4];

• *Тестируемость программы*. Код, который легче тестировать, будет содержать меньше ошибок и надежнее работать [5];

• *Возможность повторного использования частей программы*. Программу желательно проектировать так, чтобы ее модули можно было повторно использовать в других проектах [6]. Проект должен быть хорошо структурирован, не содержать дублирования, иметь хорошо оформленный код и, желательно, документацию;

• *Сопровождаемость*. Архитектура ПО должна давать возможность легко и быстро разбираться в программе [7], должна быть структурирована, не содержать дублирования, иметь хорошо оформленный код.

Главной задачей при проектировании больших систем является снижение сложности (декомпозиция). Сложная система должна строиться из небольшого количества более простых подсистем (модулей), каждая из которых, в свою очередь, должна строиться из частей меньшего размера и т. д. [8] (рис. 1).

При правильной декомпозиции программа превращается в набор модулей, взаимодействующих друг с другом по определенным правилам. Сначала программу разбивают на функциональные модули, описывающие ее функции в общем виде. Затем полученные модули анализируются более детально и, в свою очередь, делятся на подмодули либо на объекты. Декомпозицию на модули лучше всего производить, используя те задачи, которые решает программа. Основная программа разбивается на модули, которые могут выполнять возложенные на них задачи независимо друг от друга. Также при правильной декомпозиции модули могут быть сфокусированы на выполнение своих функций и независимы [9]. Модуль характеризует не только выполняемые им функции, но и набор данных, необходимый для их выполнения (рис. 2). При правильной декомпозиции модуль может выполнить свои функции лишь на основе своих входящих данных.

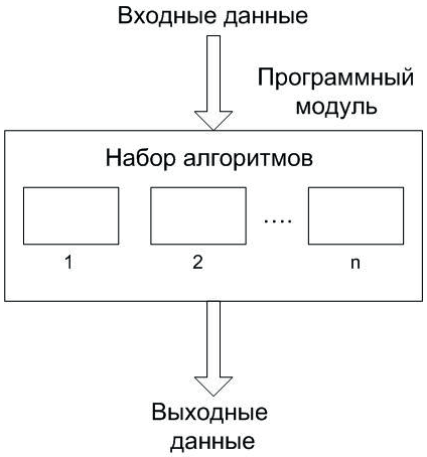

Рис. 2. Схема построения модуля

Под модульным построением чаще всего подразумевается разбиение специального программного обеспечения (СПО) на составные части (классы, функции и т. д.), каждая из которых выполняет свой спектр задач. В случае разработки программы для высоконадежных систем данного способа разбиения может оказаться недостаточно. В качестве примера высоконадежной системы можно привести измерительный комплекс сбора, хранения и анализа навигационной информации со спутников.

Выход из строя одного модуля может привести к аварийному завершению всего СПО. В связи с этим в СПО для высоконадежных систем предлагается в качестве модулей использовать отдельные программы, отвечающие за свой спектр задач. Кроме того, необходимо создать

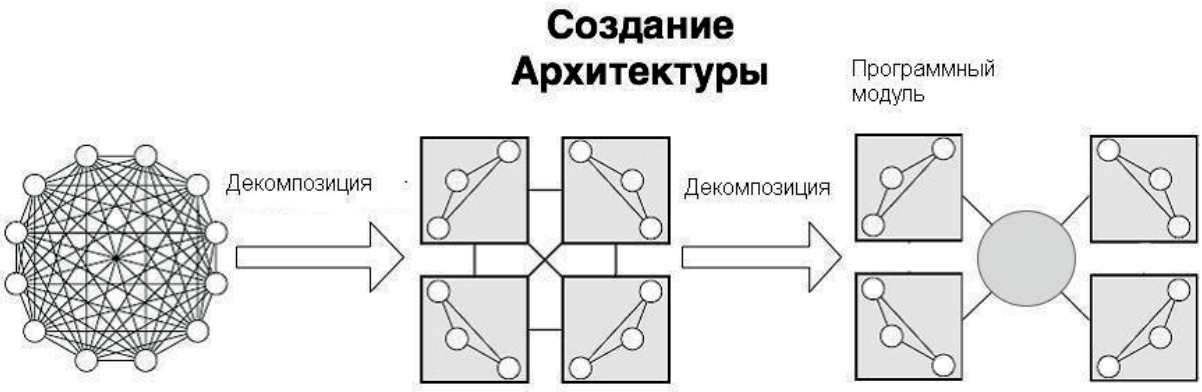

Рис. 1. Пример декомпозиции

программу посредник [10], которая будет передавать сообщения между модулями (рис. 3). В таком случае выход из строя одного из модулей не позволит выйти из строя всей системе. А произойдет только частичная потеря функционала. Для его восстановления необходимо будет перезапустить модуль. Передачу данных между программами можно организовать с помощью сетевого протокола TCP/IP, общих файлов, разделяемой памяти и т. д.

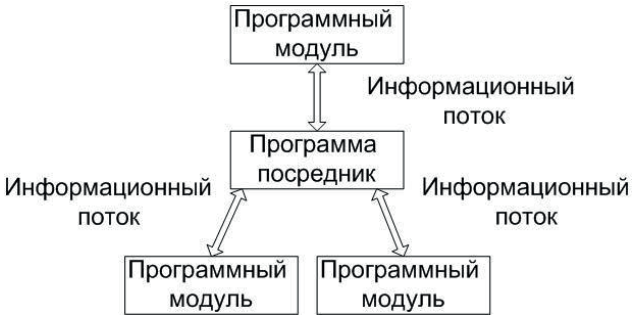

Рис. 3. Пример построения декомпозиции

В качестве примера данной технологии рассмотрим СПО, функциями которого являются передача сообщений от источника к потребителю и сохранение этой информации в базе данных. Выделим основные модули программы – это общение с источником данных, запись данных в базу и отправка данных потребителю. Выведем каждый модуль в отдельную программу, добавим в состав программу-посредник, через которую модули будут обмениваться данными (рис. 4).

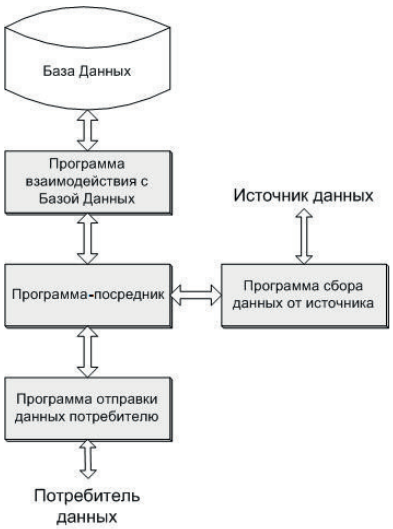

Рис. 4. Схема построения программы

Из схемы видно, что при зависании части модулей взаимодействие с базой данных продолжается за счет остальных, и при восстановлении работоспособности модуля программа будет работать в штатном режиме.

В результате проведенного вычислительного эксперимента с целью сравнения монолитной архитектуры построения ПО с предлагаемой было выявлено следующее. Во время эксперимента источник данных передал 1 000 информационных пакетов. Также была создана программа-агент, которая аварийно завершает работу тестируемой программы (в случае монолитного построения) либо одного из модулей для случая предлагаемой архитектуры. Результаты эксперимента отображены в табл. 1.

Таблица 1

Результаты эксперимента

Частота срабатывания программы-агента, с	Количество пропущенных пакетов (при монолитном построении программы), ед.	Количество пропущенных пакетов (при предлагаемой архитектуре построения программы), ед.
45	17	0
30	26	3
10	87	12
5	154	22

Как видно из табл. 1 предлагаемая архитектура построения СПО позволяет увеличить надежность разрабатываемого СПО.

Заключение

Были рассмотрены основные критерии, позволяющие оценить, насколько удачно была спроектирована архитектура программы. При проектировании ПО разработчик должен руководствоваться предлагаемыми критериями. При создании большого проекта лучше использовать архитектуру, предполагающую декомпозицию программного проекта на модули по функциональному назначению. Данный способ построения ПО позволяет упростить процесс программирования и отладки проекта. Для случая высоконадежных систем предлагается выделить каждый модуль проекта в отдельную программу. Для взаимодействия между программами следует использовать модуль (программу-посредник). Передача данных может осуществляться любым удобным для данного случая способом (например TCP/IP, каналы, общие файлы). Показано, что данная архитектура увеличивает надежность разрабатываемого ПО и имеет особый приоритет для высоконадежных систем.

Список литературы

1. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. СПб. : Питер, 2006. 575 с.
2. Гагарина Л. Г., Кокорева Е. В., Виснадул Б. Д. Технология разработки программного обеспечения : учеб. пособие. М. : Форум Инфра-М, 2013. 400 с.
3. Круз Р. Л. Структуры данных и проектирование программ : пер. с англ. М. : «БИНОМ. Лаборатория знаний», 2008. 765 с.
4. Мацяшек Л. А., Лионг Б. Л. Практическая программная инженерия на основе учебного примера : пер. с англ. М. : «БИНОМ. Лаборатория знаний», 2009. 956 с.
5. Фаулер М. Архитектура корпоративных программных приложений : пер. с англ. М. : Вильямс, 2006. 544 с.
6. Назаров С. В. Архитектура и проектирование программных систем : монография. М. : Инфра-М, 2016. 374 с.
7. Нильссон Дж. Применение DDD и шаблонов проектирования. Проблемно-ориентированное проектирование приложений с примерами на C# и .NET. М. : Вильямс, 2008. 560 с.
8. Фаулер М. Шаблоны корпоративных приложений. М. : Издат. дом «Вильямс», 2011. 544 с.
9. Руководство Microsoft по проектированию архитектуры приложений [Электронный ресурс]. URL: http://apparchguide.ms/Book (дата обращения: 12.03.2018).
10. Message Bus [Электронный ресурс]. URL: http://msdn.microsoft.com/en-us/library/ms978579.aspx (дата обращения: 15.03.2018).

История статьи
Поступила в редакцию 17 апреля 2018 г.
Принята к публикации 21 мая 2018 г.

SOFTWARE ARCHITECTURE FOR HIGHLY RELIABLE SYSTEMS

A.V. Karavanov, N. D. Ivanov
Siberian Federal University, Krasnoyarsk, Russian Federation

The problem of building software for highly reliable systems is considered in the article. The criteria that allow us to evaluate the effectiveness of software architecture were considered first. The brief description of these criteria was given. Software with a wide range of tasks is built in a modular manner. The program is divided into modules for their functional purpose. It is proposed to allocate a separate module in the program to improve the software fault tolerance in highly reliable systems. An example is given of constructing the proposed architecture with a breakdown of the software into modules (programs) for a functional purpose. For this purpose, an intermediary program has been developed, through which the modules exchange information. The interaction between the modules and the mediation program can occur through various technologies (TCP/IP, shared files, shared memory, etc.). The interaction was realized using the network protocol TCP/IP in the example. To compare the monolithically constructed program and the program made on the proposed architecture, an experiment was conducted. The data source passed packets through the built-in software. The number of delivered packages was taken as a criterion for software reliability. The software was built according to the proposed architecture. The experiment demonstrated the advantage of the built-in software.

Keywords: software architecture, software module, software decomposition, highly reliable systems.

References

1. Bass L., Clements P., Katsman R. *Arhitektura programmnogo obespecheniya na praktike* [Software architecture in practice]. St. Petersburg, Peter, 2006, 575 p. (In Russian)
2. Gagarina L. G., Kokoreva E. V., Visnadul B. D. *Tekhnologiya razrabotki programmnogo obespecheniya* [Technology of software development]. Moscow, Forum Infra-M, 2013, 400 p. (In Russian)
3. Cruz R. L. *Struktury dannyh i proektirovanie programm* [Data structures and program design]. Moscow, «BINOM. Laboratory of Knowledge», 2008, 765 p. (In Russian)
4. Matsyachek L. A., Lyong B. L. *Prakticheskaya programmnaya inzheneriya na osnove uchebnogo primera* [Practical software engineering on the basis of a case study]. Moscow, «BINOM. Laboratory of Knowledge», 2009, 956 p. (In Russian)
5. Fowler M. *Arhitektura korporativnyh programmnyh prilozhenij* [Architecture of corporate software applications]. Moscow, Williams, 2006, 544 p. (In Russian)
6. Nazarov S. V. *Arhitektura i proektirovanie programmnyh sistem* [Architecture and design of software systems]. Moscow, Infra-M, 2016, 374 p. (In Russian)

7. Nilsson J. *Primenenie DDD i shablonov proektirovaniya. Problemno-orientirovannoe proektirovanie prilozhenij s primerami na C# i .NET* [Application of DDD and design patterns. Project-oriented design of applications with examples in C # and .NET]. Moscow, Williams, 2008, 560 p. (In Russian)
8. Fowler M. *Shablony korporativnyh prilozhenij* [Corporate Application Templates]. Moscow, Publishing. house «Williams», 2011, 544 p. (In Russian)
9. Microsoft's guide to designing an application architecture. Available at: http://apparchguide.ms/Book (accessed 12.03.2018).
10. Message Bus. Available at: http://msdn.microsoft.com/en-us/library/ms978579.aspx (accessed 15.03.2018).

Article history
Received 17 April 2018
Accepted 21 May 2018