УДК 004.43:629.735.017.1.004.5

МОДУЛЬНЫЙ ПОДХОД В РАЗРАБОТКЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ МОНИТОРИНГА ЖИЗНЕННОГО ЦИКЛА КОМПОНЕНТОВ ВОЗДУШНЫХ СУДОВ

В.А. ЕРЕМИН, Г.Е. ГЛУХОВ, С.А. ПЕТРУХИН, А.Н. ШАРЫПОВ, А.Ю. КОНЬКОВ

Государственный научно-исследовательский институт гражданской авиации
г. Москва, Российская Федерация

Аннотация. В статье затронуты вопросы создания программного обеспечения на основе модульного многоязычного подхода, модульности в разработке информационной системы мониторинга летной годности воздушных судов. Причины, по которым использование нескольких языков программирования в рамках одной информационной системы может оказаться более эффективным, чем использование одного языка. Рассматривается перевернутая пирамида программного обеспечения Ола Бини, ее предметный, динамический и стабильный слой. Проблемы выбора конкретного языка в зависимости от разных факторов. Рассматривается возможная реализация модульности и многоязычности в разработке веб-ориентированной информационной системы мониторинга летной годности воздушных судов и ее возможная архитектура, использование конкретных форматов данных и протоколов для организации взаимодействия модулей, совместная работа модулей с базой данных. Особо отмечаются важные преимущества модульного подхода для сложных информационных систем, такие как: улучшение надежности и защищенности системы, значительное упрощение тестирования сущностей системы, как отдельных модулей, простая возможность расширения кода модулей и добавления новых модулей.

Ключевые слова: информационная система, модульность, многоязычность, обработка информации, база данных, летная годность, аутентичность компонентов, пирамида Бини

MODULAR APPROACH IN THE DEVELOPMENT OF AN INFORMATION SYSTEM FOR MONITORING THE LIFE CYCLE OF AIRCRAFT COMPONENTS

V.A. EREMIN, G.E. GLUKHOV, S.A. PETRUKHIN, A.N. SHARYPOV, A.YU. KONKOV

The State Scientific Research Institute of Civil Aviation, Moscow, Russian Federation

Abstract. The article deals with the issues of creating software based on a modular multilingual approach, modularity in the development of an information system for monitoring the airworthiness of aircraft. Reasons why using multiple programming languages within a single information system may be more effective than using a single language. Ola Bini's inverted software pyramid is considered, its subject, dynamic and stable layer. Problems of choosing a specific language depending on various factors. A possible implementation of modularity and multilingualism in the development of a web-based information system for monitoring the airworthiness of aircraft and its possible architecture, the use of specific data formats and protocols for organizing the interaction of modules, and the joint operation of modules with a database are considered. The important advantages of the modular approach for complex information systems are especially noted, such as: improving the reliability and security of the system, greatly simplifying the testing of system entities as separate modules, the simple ability to expand the code of modules and add new modules.

Keywords: information system, modularity, multilingualism, information processing, database, airworthiness, component authenticity, Beanie pyramid

Введение

На данный момент не создано языка программирования, который в полной мере решает все задачи информационных систем. В настоящее время существует большое количество языков, но продолжают появляться новые. Устаревшие языки не исчезают, на них написаны различные программы, и они продолжают работать. Это значит, что необходимы специалисты для развития и сопровождения существующих программ, написанных на этих языках. Вероятно, не следует ориентироваться на редко распространенные или устаревшие языки. В некоторых компаниях или организациях, в силу их специфики, могут использоваться и такие языки [1].

Текущее представление языков выглядит примерно так:

– Старые (Basic, Pascal, Fortran, Cobol, PL\I, Ada, Lisp и др.).

– Мейнстримные (PHP, C, C++, Java, C#, JavaScript, Python, Ruby и др.).

– Новые и будущие (Go, Hack, Kotlin, Rust, Scala, Swift и др.).

– Нишевые (Clojure, D, Haskell, OCaml и др.).

Существует большое количество причин, по которым создаются новые языки. Одна из главных – появление новых задач, для которых реализация решений на существующих языках либо невозможна, либо осуществляется не в полной мере [2]. Ниже перечислены несколько причин появления новых языков:

1. Несоответствие существующих языков программирования новым задачам и возросшим требованиям.
2. Недостаточность уровня абстракции в существующих языках для уровня сложности решаемых задач.
3. Необходимость избавиться от архаичного наследия «старых» языков.
4. Эволюция существующих программно-аппаратных платформ и появление новых.
5. Желание контролировать развитие языка.
6. Для проверки положений теории языков на практике или обучения.
7. Маркетинг, формирование/повышение имиджа компании, продвижение товара (например J#).
8. Энтузиазм отдельных разработчиков.

Представленная ниже таблица показывает, что, несмотря на наличие существующих языков программирования, решающих аналогичные задачи, известные IT-компании не используют готовые языки программирования, а разрабатывают собственные (табл. 1).

Многоязычные приложения, структура

Использование нескольких языков в проектах открывает новые возможности разработки программных систем. В таких системах под каждую задачу выбирается именно тот язык, средствами которого достигается наилучший результат. Эффект подхода достигается за счет использования преимуществ языка в тех частях, для которых выбранный язык будет максимально эффективным, и компенсации недостатков в тех, где лучше всего использовать другой язык [3].

Таблица 1

Новые языки от известных IT-компаний, которые появились за последние 10 лет

Компания	Год	Язык	Описание
Google	2009	Go	ориентирован на разработку сетевых многопоточных сервисов
	2011	Dart	замена JavaScript
Mozilla	2010	Rust	язык реализации алгоритмов (многоядерные архитектуры)
JetBrains (СПб, Россия)	2011	Kotlin	замена Java
RedHat	2011	Ceylon	«упрощенный» Java
Microsoft	2012	TypeScript	«улучшенный» JavaScript
Apple	2014	Swift	язык общего назначения (замена C)
Facebook	2014	Hack	замена PHP

Для понимания того, какие языки и в каком сочетании использовать в многоязычных проектах, можно рассмотреть структуру приложения по слоям: три слоя. Представим их в виде перевернутой пирамиды (рис. 1). Пояснения к рис. 1 приведены в табл. 2.

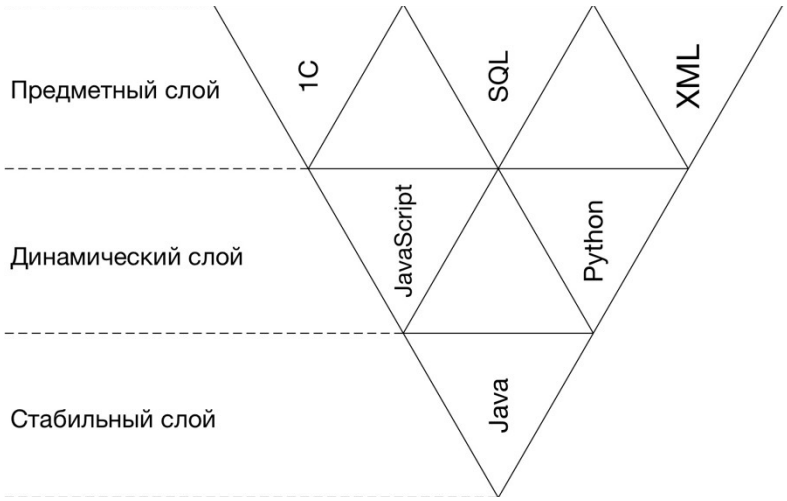

Рис. 1. Пирамида Бини

Приложения стабильного слоя, который находится на вершине перевернутой пирамиды, отвечают самым высоким требованиям по стабильности, производительности и отсутствию ошибок.

Написание программ на языках – самое дорогостоящее и трудоемкое. Неспроста пирамида является перевернутой – программное обеспечение (ПО) этого уровня существует в меньшинстве. Но затраты на создание такого ПО вполне себя оправдывают благодаря тому, что библиотеки этого слоя в значительной мере переиспользуются ПО из верхних слоев.

В динамическом слое находится ПО, которое достаточно ответственное, чтобы создавать ПО высокого уровня качества, но, одновременно, достаточно гибкое, чтобы трудоемкость создания была ниже, чем из стабильного слоя.

Верхний слой – низкий уровень ответственности, самый нестабильный. Здесь не предъявляются высокие требования к уровню знаний программиста. Вследствие чего обеспечивается большое количество разработчиков. Основное назначение данного слоя – быстрая разработка ПО. К этому слою относятся языки без контроля типов для создания ПО

типа front – end и DSL для создания прикладного ПО. Эти языки максимально приближены к своим потребителям, причем, в такой мере, что даже знаний непрограммиста может хватить для создания веб-сайта или построения базы данных для личных целей. Самый значительный объем ПО находится именно в этом слое и ему соответствует самая широкая часть пирамиды.

Таблица 2

Пояснения к пирамиде Бини

Слой	Описание	Языки
Предметный слой	Быстрая прикладная разработка	SQL, XML, XAML, 1C, веб-шаблонизаторы
Динамический слой	Гибкая, продуктивная разработка	Clojure, Python, JavaScript
Стабильный слой	Стабильный, с высокой производительностью, протестированный функционал	Java, C#, Scala

Выбор языка в проекте зависит от большого количества факторов. Ответив на некоторые вопросы, можно определиться с языком разработки:

- Насколько быстро может быть освоен язык?
- Существует ли сообщество языка и насколько оно развивается?
- Как быстро могут быть найдены нужные разработчики?
- Насколько разнообразны и эффективны существующие фреймворки?
- Насколько легко язык интегрируется в многоязычную среду?

Кратко о модульном программировании

Модульное программирование – это организация программы в виде совокупности независимых блоков, которые называются модулями. Их поведение и структура подчиняются определенным правилам. С использованием модульного программирования значительно упрощается обнаружение ошибок и тестирование программы [4].

Модульность в языках программирования – это принцип, согласно которому программное обеспечение разделяется на отдельные сущности, называемые модулями. Модуль характеризуют:

– один вход и один выход – на входе модуль получает некоторый набор исходных данных, выполняет их обработку и возвращает данные результата; реализуется принцип IPO: input – process – output, вход – процесс – выход;

– функциональная завершенность – модуль выполняет регламентированные операции для реализации каждой отдельной функции, достаточные для завершения начатой обработки;

– логическая независимость – результат работы модуля не зависит от работы других модулей, а зависит только от исходных данных.

Принципы модульного программирования программных продуктов во многом сходны с принципами нисходящего проектирования. В начале определяются состав и подчиненность функций, затем – набор программных модулей, реализующих эти функции (рис. 2) [5, 6].

Однотипные функции реализуются одними и теми же модулями. Функция верхнего уровня обеспечивается главным модулем; он управляет выполнением нижестоящих функций, которым соответствуют подчиненные модули.

Для определения набора модулей, реализующих функции какого-либо конкретного алгоритма, необходимо учитывать следующее:

- каждый модуль вызывается на выполнение вышестоящим модулем и, по окончании работы, производит возврат управления модулю, который его вызвал;

• в алгоритме принятие основных решений выносится на максимально «высокий» по иерархии уровень;

• если одна и та же функция используется в разных местах алгоритма, то создается один модуль, который вызывается по необходимости. В результате дальнейшей детализации алгоритма создается функционально-модульная схема алгоритма, которая является основой для программирования.

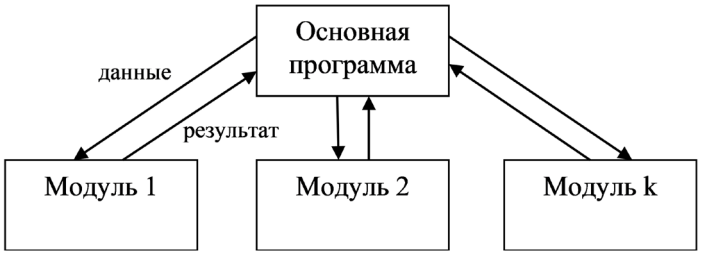

Рис. 2. Модули

Реализация многоязычности и модульности в разработке информационной системы

Общая структура информационной системы мониторинга летной годности воздушных судов представлена на рис. 3.

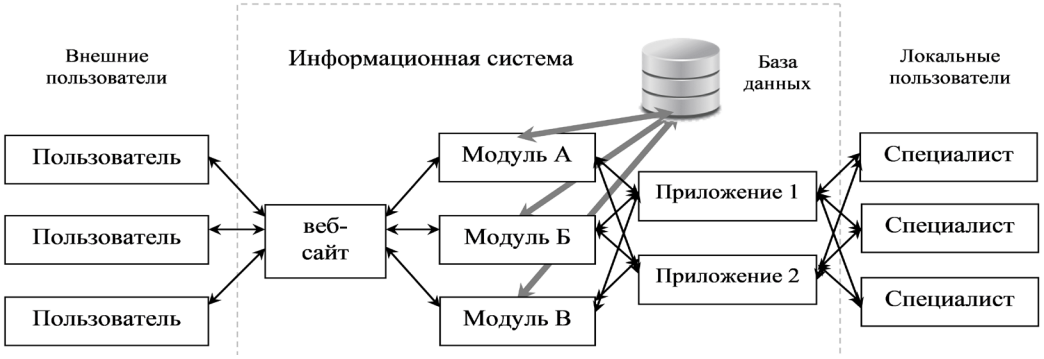

Рис. 3. Общая структура информационной системы

Веб-сайт. Веб-сайт может располагаться на сервере в дата-центре хостинговой компании или непосредственно на территории ГосНИИ ГА. По сути, представляет из себя интерактивный графический интерфейс. Внешние пользователи могут видеть информацию, хранящуюся в базе данных, фильтровать, сортировать, составлять условия для запросов и формировать различные виды отчетов. Обмен данных происходит с модулями в формате json.

Понятие JSON. JSON (JavaScript Object Notation) – это текстовый формат представления данных в нотации объекта JavaScript. JSON предназначен (наравне с другими форматами XML и YAML) для обмена данными.

Несмотря на свое название, JSON можно использовать не только в JavaScript, но и в любом другом языке программирования [7, 8].

JSON по сравнению с другими форматами обладает достаточно весьма существенным преимуществом. Он, в отличие от них, является более компактным, что очень важно при обмене данными в сети Интернет. Кроме этого, JSON более прост в использовании, его намного проще читать и писать, в сравнении с другими форматами.

Модули. Модули могут быть реализованы на разных языках программирования (например: go, python, c++). В сложившейся мировой практике для веб-систем наибольшее распространение для обмена информацией получили протокол обмена данными http и формат описания данных json. Модули работают на так называемом сервере приложений в качестве сервисов, постоянно находятся в памяти и ожидают входные данные.

Каждый модуль является автономной законченной программой, которая реализует отдельный строгий функционал. Задача модуля состоит в том, чтобы принять данные (от веб-сайта или от приложений), сделать при необходимости первичную обработку данных и отправить запрос в базу данных (рис. 4, рис. 5).

При поступлении ответа из базы данных происходит обработка полученной информации и ее отправка владельцу запроса (на веб-сайт или приложение) [9, 10].

Например, модуль А отвечает за обработку запросов по информации об аутентичности агрегатов авиакомпании.

```
POST /json/ HTTP/1.1
Host: api.xxxxxxxxxx.ru
Content-Length: 100
Content-Type: application/json; charset=utf-8

{
    "function": "GetAuthInfoBC",
    "user_id": "1",
    "ca": "COMP1",
    "bn": "12345"
}
```

Рис. 4. Пример json-запроса от внешнего источника к модулю (пользователь запрашивает данные по борту 12345, принадлежащему авиакомпании с условным кодом COMP1)

```
{
    "response_type": "data",
    "user_id": "1",
    "ca": "COMP1",
    "bn": "12345",
    "agr_num": "2",
    "agr": [
      { "zn": "11111",
        "gn": "8-1950-000",
        "dv": "1989-12-25",
        "verify_date": "2019-07-21",
        "autent": "4. Компонент ВС неутвержден
изготовителем",
      },
      { "zn": "22222",
        "gn": "КАУ-115АМ",
        "dv": "1992-10-30",
        "verify_date": "2020-03-10",
        "autent": "6. Неутвержденный по заключению
экспертов ИАЦ ГосНИИ ГА",
      }
    ],
    "order_completed": "1"
}
```

Рис. 5. Пример json-ответа от модуля к источнику запроса (найдено 2 агрегата); информация содержит данные о заводском номере агрегата, шифре, дате выпуска, дате фотодокументирования и информации об аутентичности)

База данных. База данных (хранилище информации) может быть расположена на отдельном сервере. В базу данных поступают запросы только от модулей, происходит выборка и результат поступает обратно на модули.

Приложения. Приложения для локальных пользователей могут быть как десктопными, так и веб-приложениями. Для реализации приложений в данном случае выбор языков достаточно богат.

Выводы

В сложных информационных системах находят применение многоязычные проекты с использованием модульности. Модули могут быть реализованы на разных языках и платформах. Использование модульности и многоязычности в применении к разработке программного обеспечения дает скорость, надежность, защищенность. Сочетание указанных методов способствует многократному использованию сущностей, улучшает тестируемость и расширяемость кода. В случае внесения изменений, доработок в один модуль, не затрагиваются другие модули и система в целом продолжает функционировать.

Применение описанных методов позволяет использовать единую базу данных для работы как локальных пользователей и специалистов, так и внешних, с использованием единых алгоритмов запросов и обработки информации.

ЛИТЕРАТУРА

1. Информационный интернет ресурс. URL: https://infostart.ru/1c/articles/975701/ (Дата обращения: 02.03.2021).

2. Вигерс К., Битти Дж. Разработка требований к программному обеспечению. 3-е изд., доп. Пер. с англ. М.: Русская редакция; СПб.: БХВ-Петербург, 2014. 736 с.

3. Фаулер М. Предметно-ориентированные языки программирования. Пер. с англ. М.: ООО «ИД Вильямс», 2011. 576 с.

4. Эванс Э. Предметно-ориентированное проектирование. Структуризация сложных программных систем. Пер. с англ. М.: ООО «И.Д. Вильямс», 2017. 448 с.

5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного программирования. Паттерны проектирования. СПб: Питер, 2015. 368 с.

6. Фримен Э., Робсон Э. Паттерны проектирования. Обновленное юбилейное издание. СПб.: Питер, 2018. 656 с.

7. Эванс Б., Вербург М. Java. Новое поколение разработки. СПб.: Питер, 2014. 560 с.

8. Петрухин С.А.,. Глухов Г.Е., Ладыгина Н.Н. Двоичная классификация авиационных текстов с использованием нейронной сети // Научный вестник ГосНИИ ГА. 2021. № 34. С. 50-58.

9. Карапетян А.Г., Черников П.Е., Гаранин С.А., Брусникин В.Ю., Коваль С.В., Быкова В.В. Организация системы доступа и безопасности Центральной нормативно-методической библиотеки гражданской авиации // Научный вестник ГосНИИ ГА. 2021. № 34. С. 91–101.

10. Brusnikin V. Yu., Sharypov A.N., Glukhov G.E., Karapetyan A.G., Koval S.V., Chernikov P. E. Aircraft components life cycle monitoring system as an element of the state control of aviation equipment operation maintenance. Test Engineering and Management/ Mattingley Publishing Co., Inc. ISSN:0193–4120. Pp. 14 535–14 545.

REFERENCES

1. Internet information resource. Available at: https://infostart.ru/1c/articles/975701/ (Accessed: 02.03.2021).

2. Wiegers K., Beatty J. *Razrabotka trebovanij k programmnomu obespecheniyu* [Software requirements]. Third Edition. Moscow. Russkaya redaktsiya Publ., St. Petersburg. BHV-Petersburg Publ., 2014, 736 p. (In Russian).

3. Fowler M. *Predmetno-orientirovannye yazyki programmirovaniya* [Subject-oriented programming languages]. Moscow, Williams Publishing House, 2011, 576 p. (In Russian).

4. Evans E. *Predmetno-orientirovannoe proektirovanie. Strukturizatsiya slozhnykh programmnykh system* [Subject-oriented design. Restructuring Complex Software Systems]. Moscow, Williams Publishing House, 2017, 448 p. (In Russian).

5. Gamma E., Helm R., Johnson R., Vlissides J. *Priemy ob`ektno-orientirovannogo programmirovaniya. Patterny proektirovaniya* [Methods of object-oriented programming. Design Patterns]. St. Petersburg, Piter Publ., 2015, 368 p. (In Russian).

6. Freeman E., Robson E. *Patterny proektirovaniya* [Design patterns]. Updated anniversary edition. St. Petersburg, Piter Publ., 2018, 656 p. (In Russian).

7. Evans B., Verburg M. *Java. Novoe pokolenie razrabotki* [Java. A new generation of development]. St. Petersburg, Piter Publ., 2014, 560 p. (In Russian).

8. Petrukhin S.A., Glukhov G.E., Ladygina N.N. Binary classification of aviation texts using a neural network. *Nauchnyj vestnik GosNII GA = Scientific Bulletin of The State Scientific Research Institute of Civil Aviation,* 2021, no. 34, pp. 50-58. (In Russian).

9. Karapetyan A.G., Chernikov P.E., Garanin S.A., Brusnikin V.Yu., Koval S.V., Bykova V.V. Organization of the access and security system of the central normative-methodical library of civil aviation. *Nauchnyj vestnik GosNII GA = Scientific Bulletin of The State Scientific Research Institute of Civil Aviation,* 2021, no. 34, pp. 91–101. (In Russian).

10. Brusnikin V. Yu., Sharypov A.N., Glukhov G.E., Karapetyan A.G., Koval S.V., Chernikov P. E. Aircraft components life cycle monitoring system as an element of the state control of aviation equipment operation maintenance. Test Engineering and Management/ Mattingley Publishing Co., Inc. ISSN:0193–4120, pp. 14 535–14 545.

СВЕДЕНИЯ ОБ АВТОРАХ

Еремин Владимир Александрович, ведущий инженер Информационно-аналитического центра, ФГУП Государственный научно-исследовательский институт гражданской авиации, ул. Михалковская, 67, корпус 1, Москва, Российская Федерация, 125438; e-mail: eremin@mlgvs.ru.

Глухов Геннадий Евгеньевич, заместитель директора Центра по информационным технологиям, ФГУП Государственный научно-исследовательский институт гражданской авиации, ул. Михалковская, 67, корпус 1, Москва, Российская Федерация, 125438; e-mail: glukhov@mlgvs.ru.

Петрухин Сергей Александрович, старший инженер Информационно-аналитического центра, ФГУП Государственный научно-исследовательский институт гражданской авиации, ул. Михалковская, 67, корпус 1, Москва, Российская Федерация, 125438; e-mail: petruhin@mlgvs.ru.

Шарыпов Андрей Николаевич, заместитель директора центра, ФГУП Государственный научно-исследовательский институт гражданской авиации, ул. Михалковская, 67, корпус 1, Москва, Российская Федерация, 125438; e-mail: sharypov@mlgvs.ru.

Коньков Александр Юрьевич, заместитель начальника отдела Информационно-аналитического центра, ФГУП Государственный научно-исследовательский институт гражданской авиации, ул. Михалковская, 67, корпус 1, Москва, Российская Федерация, 125438; e-mail: konkov@mlgvs.ru.

ABOUT THE AUTHORS

Eremin Vladimir A., Lead Engineer of Center, The State Scientific Research Institute of Civil Aviation, Mikhalkovskaya Street, 67, building 1, 125348 Moscow, Russian Federation; e-mail: eremin@mlgvs.ru.

Glukhov Gennady E., Deputy Director of the Center for Information Technology, The State Scientific Research Institute of Civil Aviation, Mikhalkovskaya Street, 67, Building 1, 125438 Moscow, Russian Federation; e-mail: glukhov@mlgvs.ru.

Petrukhin Sergey A., Senior Engineer of Center, The State Scientific Research Institute of Civil Aviation, Mikhalkovskaya Street, 67, Building 1, 125438 Moscow, Russian Federation; e-mail: petruhin@mlgvs.ru.

Sharypov Andrey N., Deputy Director of the Scientific Center, The State Scientific Research Institute of Civil Aviation, Mikhalkovskaya Street, 67, building 1, 125438 Moscow, Russian Federation; e-mail: sharypov@mlgvs.ru.

Konkov Alexander Yu., Deputy Head of Department of the Scientific Center, The State Scientific Research Institute of Civil Aviation, Mikhalkovskaya Street, 67, building 1, 125438 Moscow, Russian Federation; e-mail: konkov@mlgvs.ru.