Инженерный вестник Дона, №6 (2023)
ivdon.ru/ru/magazine/archive/n6y2023/8481

Сравнение гибкой методики разработки CRM-систем с антипаттерном «hard coding»

В.А. Шуринова, С.Г. Симагина

Поволжский государственный университет телекоммуникаций и информатики, Самара

Аннотация: Разработка программного обеспечения, а именно - CRM-системы (Customer Relationship Management) для повышения эффективности работы компании, вне зависимости от сферы, является наиболее эффективным решением для бизнеса в части организационно-управленческой деятельности. Важным аспектом успешной реализации и внедрения системы в компанию, является принцип разработки и построения архитектуры системы на серверном уровне. Существует несколько подходов в области применения средств разработки для настройки функциональности в системах, однако не всегда конечный результат удовлетворяет бизнес-потребность клиента. Корректность работы системы основана на важном факторе – дальнейшая поддержка существующего кода, данное требование актуально для любого проекта и зависит от изначально выбранного метода разработки системы и качества задач, выполненных программистами.
Ключевые слова: CRM-система, BPM, hardcode, разработка, гибкая настройка.

Введение

В настоящее время разработано и используется большое количество решений в области автоматизации, многие из которых являются интегрированными системами управления, помогающими облегчить, ускорить и сократить время проведения операций [1, 2]. Например, одними из самых востребованных на рынке РФ остаются системы класса CRM (Customer Relationship Management) на базе BPM (Business Process Management System). Сегодня в BPM-системах заинтересованы компании разных масштабов и сфер деятельности: производство, госсектор, энергосбыт и ЖКХ, недвижимость, различные услуги населению, финансы и ритейл. Компании нацелены на качественное и быстрое управление взаимоотношениями с клиентами, а также на масштабируемость решений, возможность внести изменения в готовый функционал системы, в случае необходимости, и в короткие сроки адаптировать его под текущие требования пользователей. CRM-системы на базе BPM в большей степени ориентированы на low-code среду разработки, однако, в большинстве

случаев, для крупных компаний необходима специфика, которую невозможно реализовать на этом слое и разработку необходимо передавать на уровень ниже – серверный [3, 4]. На проектах внедрения BPM-платформ в крупных компаниях также часто используют код на языке типа javaSAFEscript для разработки определенного функционала, который в полной мере не может быть реализован только средствами конструктора BPM [5].

Так как со временем бизнес-процессы компании могут меняться, а также могут меняться и локальные данные, с которыми работают сотрудники компании, внедряемая CRM-система должна быть реализована достаточно гибко, чтобы в последующем при работе в системе у пользователей не возникли дополнительные трудности, и адаптация к изменениям прошла «бесшовно». При разработке определенной функции в системе, программисты часто совершают распространенную ошибку, которая заключается в «принудительном» присвоении переменной какого-либо значения, вместо того чтобы присваивать его динамически, в зависимости от ситуации, и делать функционал более гибким. Такой принцип написания кода имеет распространенное название – hardcode [6]. Жесткий код имеет одну цель, и заключается в том, чтобы гарантировать, что значение в жестком коде не может быть изменено во время выполнения программы, однако такой подход в последующем может привести к ситуации, когда прописанная константа в коде должна быть другой. В таком случае, необходимо привлекать технических специалистов – разработчиков, для изменения значения на уровне кода, а это дополнительные временные и денежные затраты.

В рамках реализации системы на базе BPM есть вариант разработки функционала как методом low-code, так и на серверном уровне с помощью кода – это зависит от того, что именно необходимо настроить в системе.

Рассмотрим варианты настройки функционала печатных форм для банка на базе популярной CRM - системы BPMSoft. Требование от заказчика заключается в том, чтобы на существующей странице обращения клиента, который является юридическим лицом, была доступна кнопка для автоматического формирования документа - анкеты с данными о компании-клиенте. Ниже приведены примеры уже реализованной кнопки «Формирование анкеты» (рис.1), при нажатии на которую, доступны для выбора документы для автоматического формирования.

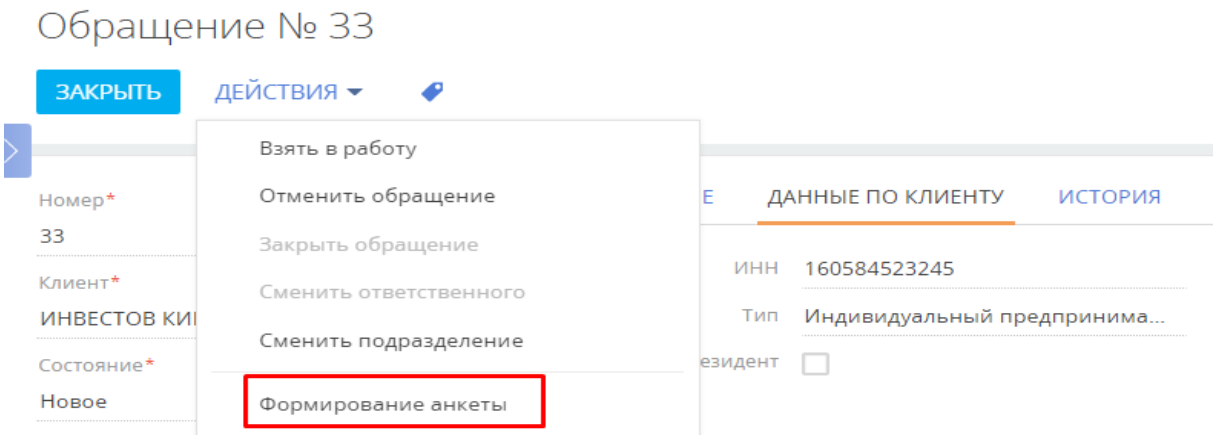

Рис. 1. – Настройка кнопки для формирования анкеты клиента

Набор доступных документов зависит от типа клиента, по которому сформированы обращения в системе, например, для клиента – индивидуального предпринимателя доступна форма «Анкета ИП», однако для клиента – юридического лица данная форма недоступна, но есть возможность сформировать документ «Анкета ЮЛ» (рис.2).

Данная настройка осуществляется с помощью средств разработки, доступность и отображение кнопкой также настраивается на уровне кода и возможность кастомизации с помощью инструментов low-code в данном случае не предусматривается.

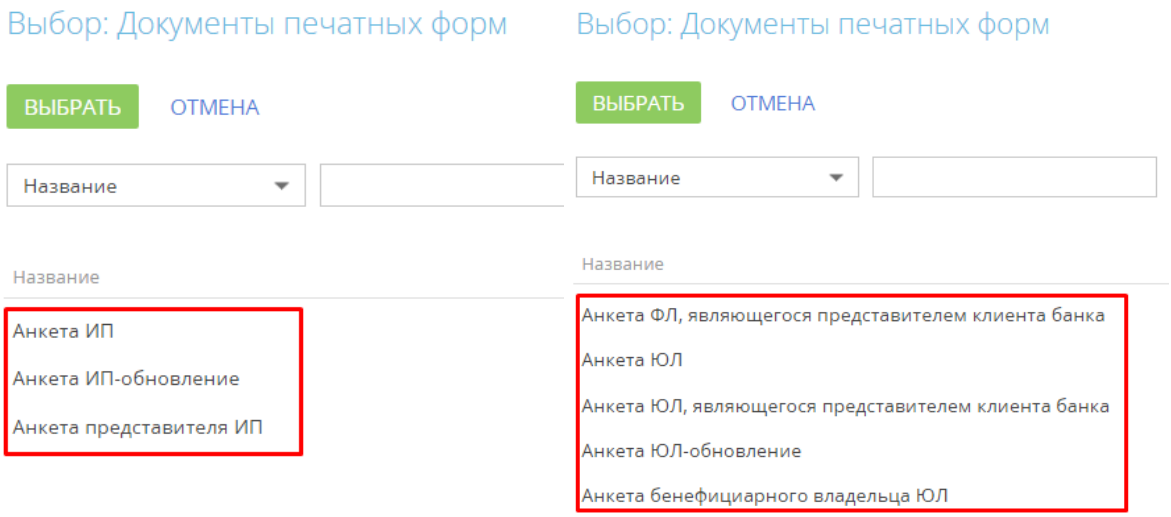

Рис. 2. – Доступные документы печатных форм относительно типа клиента

Отображение доступных документов для формирования в данном случае может быть произведено двумя способами:

1. Настройка фильтрации отображения доступных печатных форм путем изменения исходного кода – для определенного типа клиента (ИП или ЮЛ) прописываются доступные печатные формы.

2. Гибкая настройка фильтрации отображения доступных печатных форм для формирования документов на уровне кода, через связанные справочные объекты, которые настраиваются в самой системе BPMSoft [7, 8].

Рассмотрим более подробно первый вариант реализации, его плюсы и минусы. Это настройка отображения документов для выбора, относительно типа клиента с помощью средств разработки, когда для определенного типа клиента будет прописаны набор доступных для выбора печатных форм документов. Такой вариант реализации подразумевает жесткое кодирование (hard coding) [9]. Данный антипаттерн разработки достаточно распространен и в дальнейшем при работе с системой может привести к одной из ключевых проблем – реализованный код будет корректно работать только с той логикой, под которую он сделан и в случае изменения каких-либо бизнес-процессов компании, появится необходимость переписывать код под новые условия работы системы.

Например, в коде жестко будет прописано, что для типа клиента, который является индивидуальным предпринимателем, необходимо отображать документ для выбора «Анкета ИП», а если клиент является юридическим лицом, то отображать документ «Анкета ЮЛ». В данном случае, система будет работать только так и никак иначе, но, если в будущем, банк будет расширять свой спектр услуг и начнет принимать обращения от клиентов, которые занимаются частной практикой, и не являются индивидуальным предпринимателем или юридическим лицом, то формирование в системе документов для таких клиентов банка будет недоступно и для внесения доработок в систему необходимо будет вносить изменения в код программы.

Основные плюсы подхода hard-code:

- Технически такое решение занимает меньше строк кода, задача на настройку выполняется в разы быстрее и стоит дешевле.
- Работоспособность системы в данном случае будет корректной, но только при определённых условиях.

Основные минусы подхода hard-code:

- Могут проявляется непредсказуемые дефекты во время переноса, переименования файлов, и их поведение может меняться при изменении конфигурации устройств.
- Невозможность гибкой настройки.
- Усложняет тестирование системы.

Вторым вариантом реализации настройки отображения доступных документов печатных форм в BPM-системе является гибкая настройка на уровне кода с помощью дополнительной реализации связанных объектов в системе. Такой подход поможет в дальнейшем, при возникновении каких-либо изменений в процессах компании, адаптировать функционал системы

под нужные условия работы без дополнительных трудозатрат и внесения изменений в код программы.

Для гибкой настройки в системе необходимо реализовать справочный объект, который будет связывать тип клиента, по которому сформировано обращение в системе и доступные печатные формы для данного типа. Для настройки такого объекта в системе изначально должны быть настроены отдельные объекты, в которых аккумулированы тип клиентов и все печатные формы документов, с которыми работают пользователи системы, относительно страницы обращения клиента.

Таким образом, связывающий справочный объект должен содержать следующие связи значений (табл.1),

Таблица № 1

Документы печатных форм по типу клиента

Документы печатных форм	**Тип клиента**
Анкета ЮЛ	Юридическое лицо
Анкета ЮЛ, являющегося представителем клиента банка	Юридическое лицо
Анкета бенефициарного владельца ЮЛ	Юридическое лицо
Анкета ИП	Индивидуальный предприниматель
Анкета представителя ИП	Индивидуальный предприниматель

Определение доступности документов для последующего формирования печатной формы со страницы обращения разработчик описывает на уровне кода, но использует для фильтрации данных именно соотношение, которое прописано в новом связывающем объекте. То есть, относительно типа клиента, пользователю в системе будет отображены все документы, которые относятся к данному типу.

В случае, если в бизнес-процессах банка, сотрудники которого работают в системе, произойдут изменения и, например, у пользователя должна быть возможность формирования печатной формы документа, для

клиента с типом «Частная практика», то системный администратор, который ответственен за поддержку системы и имеет доступ к редактированию справочных объектов (частая практика в крупных компаниях, сотрудники которой работают в CRM-системах) [10], сможет добавить новую запись в связывающий справочник, а она будет определять, какие документы доступны для формирования в системе со страницы обращения для типа клиента «Частная практика».

При использовании данного подхода потребности изменения настройки на уровне кода нет. Масштабируемость настройки в данном случае не зависит от каких-либо других факторов, и если какая-либо из существующих печатных форм уже не актуальна, и не используется в системе при работе с обращением клиента, то ее можно удалить из общего справочного объекта и таким образом она более не будет отображена для выбора пользователю.

Итак, можно выделить основные плюсы данного подхода с применением гибкой настройки функциональности в системе через связывающие объекты:

- Быстрая и простая адаптация функциональности системы под изменения бизнес-процессов компании.
- Масштабируемость решений, в рамках которых с помощью внутренних инструментов системы и изначально корректно написанного кода, можно реализовать более сложную функциональность.
- Уменьшение вероятности возникновения ошибок при изменении конфигураций устройств, так как все объекты созданы на уровне системы.
- Более оптимизированное тестирование реализованной функциональности.

- Дальнейшее сопровождение и доработки существующей функциональности проходит быстрее и без рисков возникновения сопутствующих ошибок.

Однако, есть и минусы данного решения при настройке системы, а именно - сроки и сумма реализации. Так как для гибкой настройки какой-либо функциональности в системе необходимы связывающие справочные объекты, то на их реализацию также потребно дополнительное время, соответственно, и финансовые затраты при выборе такого подхода могут быть выше.

Использование принципа гибкой методики разработки CRM-систем является наиболее актуальным и корректным решением, относительно принципа hard coding. Данный вывод можно сделать, исходя из приведенного примера настройки отображения доступных для выбора печатных форм, относительно типа клиента, по данным которого должен быть сформирован документ в системе. Реализация логики функциональности с помощью использования связанных объектов в системе имеет больше плюсов и наиболее корректна по части архитектуры системы.

Литература

1. Бахтинова Ч.О., Чунаева М.Э. Автоматизация системы контроля качества при организации строительства особо опасных и технически сложных объектов в России // Инженерный вестник Дона, 2022, №3. URL: ivdon.ru/ru/magazine/archive/n3y2022/7511.

2. Башлы П.Н., Адамова О.В. Автоматизация и управление технологическими процессами перспективного пункта пропуска // Инженерный вестник Дона, 2021, №6. URL: ivdon.ru/ru/magazine/archive/n6y2021/7022.

3. Резник Г.А., Парамонова Л.С. Внедрение CRM-системы на производственно-коммерческом предприятии: основные этапы и шаги // Экономика и социум. 2017. № 4(35). С. 1153-1158.

4. Kale V. Enhancing Enterprise Intelligence: Leveraging ERP, CRM, SCM, PLM, BPM, and BI. CRC Press, 2016. 382 c.

5. Истомин К. No-, low-, hard-code. Где остановиться? // URL: ecm-journal.ru/material/no-_low-_hard-code_gde_ostanovitsja (дата обращения: 17.05.2023)

6. Полищук А. Антипаттерны в программировании и проектировании архитектуры // URL: bool.dev/blog/detail/antipatterny-v-programmirovanii-i-proektirovanii-arkhitektury (дата обращения: 17.05.2023)

7. Андирякова О., Крюкова А., Иваев М. Применение low-code технологии для решения бизнес-задач // Индустриальная экономика. 2023. № 2. С. 20-24.

8. Зайков В.П., Бутмерчук Е.Б. Формирование модели бизнес-процесса в целях интеграции программных компонентнов информационной системы предприятия // Развитие современной науки и образования: актуальные вопросы, достижения и инновации: сборник статей II Международной научно-практической конференции, 23 марта 2022, Пенза. Пенза: МЦНС «Наука и просвещение», 2022. С. 30-33.

9. Барабанов А., Макрушин Д. Аутентификация и авторизация в микросервисных приложениях: обзор архитектурных подходов // Вопросы кибербезопасности. 2020. № 4(38). С. 32-43.

10. Кашковский А. Сравнение Creatio и Hubspot // URL: vc.ru/services/282904-sravnenie-creatio-i-hubspot (дата обращения: 17.05.2023)

References

1. Bahtinova Ch.O., Chunaeva M.Je. Inzhenernyj vestnik Dona, 2022, №3. URL: ivdon.ru/ru/magazine/archive/n3y2022/7511.

2. Bashly P.N., Adamova O.V. Inzhenernyj vestnik Dona, 2021, №6. URL: ivdon.ru/ru/magazine/archive/n6y2021/7022.

3. Reznik G.A., Paramonova L.S. Jekonomika i socium. 2017. №4 (35). pp. 1153-1158.

4. Kale V. Enhancing Enterprise Intelligence: Leveraging ERP, CRM, SCM, PLM, BPM, and BI. CRC Press, 2016. 382 p.

5. Istomin K. No-, low-, hard-code. Gde ostanovit'sja? [No-, low-, hard-code. Where to stay?] URL: ecm-journal.ru/material/no-_low-_hard-code_gde_ostanovitsja (accessed: 17.05.2023)

6. Polishhuk A. Antipatterny v programmirovanii i proektirovanii arhitektury [Antipatterns in programming and architecture design]. URL: bool.dev/blog/detail/antipatterny-v-programmirovanii-i-proektirovanii-arkhitektury (accessed: 17.05.2023)

7. Andirjakova O., Krjukova A., Ivaev M. Industrial'naja jekonomika. 2023. № 2. pp. 20-24.

8. Zajkov V.P., Butmerchuk E.B. Razvitie sovremennoj nauki i obrazovanija: aktual'nye voprosy, dostizhenija i innovacii: sbornik statej II Mezhdunarodnoj nauchno-prakticheskoj konferencii, 23 marta 2022, Penza. Penza: MCNS “Nauka i prosveshhenie”, 2022. pp. 30-33.

9. Barabanov A., Makrushin D. Voprosy kiberbezopasnosti. 2020. №4(38). pp. 32-43.

10. Kashkovskij A. Sravnenie Creatio i Hubspot [Comparison of Creatio and Hubspot]. URL: vc.ru/services/282904-sravnenie-creatio-i-hubspot (accessed: 17.05.2023)