Некоторые аспекты создания инструментария биллинга на инфраструктурных предприятиях на базе 1с:Предприятие

Лупанов Никита Владимирович
аспирант, Российский экономический университет имени Г.В. Плеханова, lupanov.nik2020@mail.ru

Статья посвящена созданию тиражируемого инструментария биллинга для инфраструктурных предприятий. Под биллингом понимается автоматизированный расчёт стоимости оказанных услуг, в качестве примера рассматриваются операторы морских терминалов. Когда одна и та же биллинговая система на платформе «1С:Предприятие» внедряется у нескольких заказчиков, клиентские доработки постепенно смешиваются с общей логикой расчёта. В итоге у каждого заказчика появляется своя копия решения, и затраты на сопровождение растут вместе с числом внедрений. Цель исследования состоит в разработке модели модульного построения и сопровождения биллинговой системы, позволяющей тиражировать решение без клиентских копий ядра. Предложена продуктовая модель из четырёх классов компонентов: ядро, функциональные пакеты, адаптеры конфигурации-носителя и проектные расширения. Формализованы правило отнесения функциональности к этим классам и ограничения на граф зависимостей. Описан инструментарий хранения, версионирования и поставки на основе 1С:EDT и Git, включая модель совместимости состава поставки. Экономическая модель затрат на распространение изменений позволяет найти число внедрений, начиная с которого модульное сопровождение выгоднее копирования. Модель апробирована на 98 объектах метаданных действующей конфигурации.

Ключевые слова: биллинг, тарификация услуг, инфраструктурное предприятие, оператор морского терминала, инструментальные методы экономики, модульная архитектура, 1С:Предприятие, 1С:EDT, управление версиями, экономическая эффективность информационных систем.

Введение

Под инфраструктурным предприятием в работе понимается организация, которая оказывает большому числу потребителей услуги по использованию капиталоёмкой инфраструктуры (транспортной, энергетической, коммунальной или телекоммуникационной) и получает за них плату по установленным тарифам. Круг таких организаций во многом совпадает со сферами деятельности субъектов естественных монополий из ст. 4 Федерального закона «О естественных монополиях»: услуги в транспортных терминалах, портах и аэропортах, железнодорожные перевозки, передача электрической и тепловой энергии, связь, водоснабжение [1]. Сюда же относятся операторы морских терминалов, которые занимаются перевалкой и хранением грузов [2].

От торговых и производственных организаций такие предприятия отличаются прежде всего структурой выручки. Она складывается из начислений за множество однотипных операций, и каждая операция оценивается по своему тарифу с учётом договора, вида груза, срока и объёма услуги. Факт операции фиксируется в операционной системе (у морского терминала это система управления терминалом), финансовый результат отражается в учётной системе, а связывает их биллинг, то есть автоматизированный расчёт стоимости оказанных услуг. В электросвязи, например, требования к таким системам закреплены нормативно [3]. Ошибка или задержка расчёта сразу сказывается на выручке и дебиторской задолженности, поэтому биллинг следует рассматривать как часть экономической информационной системы предприятия и инструмент управления [9; 10].

В исследованиях цифровой трансформации морских портов основное внимание уделяется операционным процессам: автоматизации и роботизации терминалов, интеллектуальной инфраструктуре, имитационному моделированию контейнерных терминалов [8; 13]. Системам расчёта за услуги, которые можно тиражировать между разными операторами, посвящено заметно меньше работ.

На российском рынке биллинг для терминальных операторов часто строится на платформе «1С:Предприятие» как расширение типовой учётной конфигурации, например «1С:Управление торговлей» (далее конфигурация-носитель). При внедрении у нескольких заказчиков доработки под клиента нередко вносятся прямо в общий код, и со временем у каждого заказчика появляется своя копия системы. Любое исправление приходится вручную переносить в каждую копию, причём перенос часто занимает больше времени, чем само исправление: сначала нужно понять, что в этой копии уже изменено под клиента. Копии расходятся, и стоимость сопровождения растёт вместе с числом внедрений. В программной инженерии это называют накоплением технического долга [16], в экономике речь идёт о росте совокупной стоимости владения решением.

Работа обобщает практический опыт проектирования архитектуры тиражируемой системы биллинга для операторов морских терминалов на платформе «1С:Предприятие» и подготовки её перехода на среду разработки 1С:EDT и систему контроля версий Git. Предлагаемая модель сложилась в ходе этой работы. Статья продолжает цикл исследований автора, посвящённых инструментарию цифровой трансформации и поддержке принятия решений в инфраструктурных предприятиях [5; 6].

Цель исследования состоит в разработке модели модульного построения и сопровождения биллинговой системы инфраструктурного предприятия, которая позволяет тиражировать решение без клиентских копий его ядра. Для этого нужно определить состав компонентов решения, формализовать правила отнесения функциональности к ним и допустимые зависимости, описать инструментарий хранения, версионирования и поставки, оценить экономический эффект и проверить модель на действующей конфигурации.

Научная новизна работы заключается в формализации продуктовой модели биллинговой системы через отображение классификации и ограничения на граф зависимостей её объектов, а также в пороговом условии экономической эффективности модульного сопровождения в зависимости от числа внедрений.

Продуктовая модель тиражируемого биллингового решения

Биллинг терминального оператора хранит тарифы и договорные условия (типы и варианты цен, квоты, публичные оферты, правила округления), рассчитывает стоимость оказанных услуг по операционным событиям, фиксирует начисления и передаёт их в учётный контур для оформления реализации. Эти функции неоднородны. Одни нужны любому терминалу, другие только терминалам определённой специализации, третьи появляются из-за устройства конкретной учётной конфигурации, и лишь небольшая часть отражает уникальные условия отдельного клиента. Отсюда четыре класса компонентов продуктовой модели (рис. 1):

ядро: обязательные сценарии, без которых продукт нельзя считать законченным (тарификация, договорные правила, калькуляция, генеральный груз, регистр начислений, журнал событий);

функциональные пакеты: сценарии, нужные группе заказчиков (контейнеры, железнодорожные операции, RO-RO, наливные и навалочные грузы, предварительный расчёт);

адаптеры конфигурации-носителя: перевод контрактов ядра в объекты конкретной учётной конфигурации (договоры, контрагенты, номенклатура, реализация);

проектные расширения: доработки конкретного клиента, которые другим заказчикам не нужны.

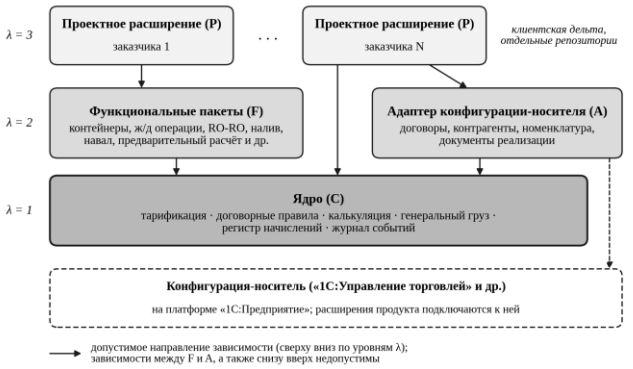


Рис. 1. Продуктовая модель тиражируемой системы биллинга Источник: составлено автором.

В модели важны два положения. Ядро у продукта одно, и варианты ядра под отдельную учётную конфигурацию или отдельного клиента не допускаются. Всё, что связано с конфигурацией-носителем, собирается в адаптере, даже если функцией пользуются все заказчики на этой конфигурации. Например, автоматическое создание заказа клиента по результатам калькуляции нужно из-за устройства «1С:Управление торговлей», поэтому относится к адаптеру. Тогда к тому же ядру можно подключить другую учётную конфигурацию без копирования. Такое разделение соответствует подходу линеек программных продуктов [14; 18] и принципу, согласно которому устойчивые компоненты не должны зависеть от изменчивых [7; 17].

Формализация правил классификации и зависимостей

Обозначим через $O = \{o_1, \dots, o_n\}$ множество объектов метаданных решения (справочников, документов, регистров, обработок), а через $K = \{C, F, A, P\}$ множество классов компонентов: ядро (C), функциональный пакет (F), адаптер носителя (A) и проектное расширение (P). Архитектурная классификация задаётся отображением

$$\kappa: O \to K. \qquad (1)$$

Для каждого объекта и для каждого нового требования до начала работ по нему проверяются три признака. Признак $a(o)$ истинен, если логика обусловлена устройством конкретной конфигурации-носителя, $c(o)$ означает, что сценарий входит в обязательную часть продукта, $f(o)$ показывает, что сценарий нужен группе заказчиков. Правило классификации выглядит так:

$$\kappa(o) = \begin{cases} A, & a(o); \\ C, & \neg a(o) \wedge c(o); \\ F, & \neg a(o) \wedge \neg c(o) \wedge f(o); \\ P & \text{в остальных случаях.} \end{cases} \qquad (2)$$

Признак $a(o)$ проверяется первым, иначе в ядро постепенно попадает код, специфичный для одной учётной конфигурации.

Связи между объектами описываются ориентированным графом зависимостей $G = (O, E)$, где $(u, v) \in E$, если объект u использует объект v. Каждому классу присвоим уровень $\lambda: K \to \mathbb{N}$: $\lambda(C) = 1$, $\lambda(F) = \lambda(A) = 2$, $\lambda(P) = 3$, и обозначим $\ell(o) = \lambda\big(\kappa(o)\big)$. Зависимость допустима, если объекты относятся к одному классу либо зависимый объект находится на более высоком уровне. Множество недопустимых зависимостей:

$$V = \{(u, v) \in E: \kappa(u) \neq \kappa(v),\ \ell(u) \leq \ell(v)\}. \qquad (3)$$

Архитектура корректна, если $V = \varnothing$. Из условия (3) следует, что ядро не зависит от пакетов, адаптеров и клиентского кода, а пакеты и адаптеры не зависят друг от друга, то есть функциональный пакет ничего не «знает» о конкретной учётной конфигурации. Изменение, добавляющее элемент в V, возвращается на перепроектирование.

Отдельный вопрос связан с физической формой пакета: оставить его в основном расширении продукта или выделить в самостоятельное расширение. Для пакета $F_j \subseteq O$ обозначим через E_j его зависимости от объектов ядра, а через $E_j^{int} \subseteq E_j$ зависимости от объектов вне публичного интерфейса ядра. Связанность пакета с ядром оценивается коэффициентом

$$k_j = \frac{\left|E_j^{int}\right|}{|E_j|}. \qquad (4)$$

Пакет стоит выделять, если k_j ниже установленного порога, он может отсутствовать у части заказчиков и тестируется независимо. Иначе он поставляется вместе с ядром. Так решение о физической модульности опирается на измеримые признаки, а не на название функции.

Инструментарий хранения, версионирования и поставки

В среде разработки 1С:EDT каждое расширение конфигурации оформляется отдельным проектом, связанным с базовым проектом конфигурации-носителя [4; 12]. Ядро и тесно связанные пакеты размещаются в основном расширении продукта, остальные компоненты выносятся отдельно только при выполнении описанных выше условий. Окончательное число расширений определяется после импорта действующей конфигурации в 1С:EDT и построения фактического графа зависимостей.

Исходный код продукта хранится в едином репозитории Git (ядро, пакеты, адаптеры, манифесты, тесты, документация), проектные расширения заказчиков размещаются в отдельных репозиториях. Клиентский репозиторий не содержит копии ядра и ссылается на выпущенную версию продукта, а внешний разработчик по умолчанию получает доступ только к клиентской дельте.

Ветвление построено на упрощённой схеме Git Flow [15] (рис. 2). Ветки описывают изменение решения во времени, а не его варианты, поэтому долгоживущих веток под учётную конфигурацию или функциональный пакет нет: по сути они воссоздали бы те же клиентские копии.

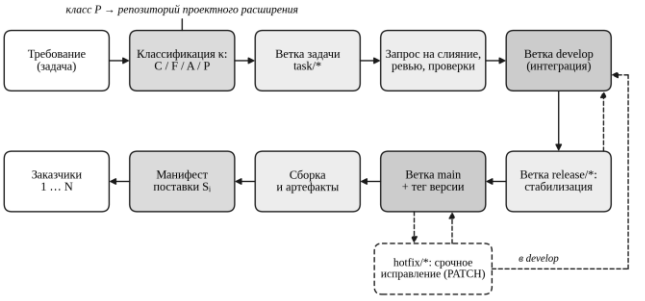


Рис. 2. Жизненный цикл изменения и формирование поставки Источник: составлено автором.

Версии назначаются по правилам семантического версионирования MAJOR.MINOR.PATCH [19]. Состав поставки заказчика i фиксируется в манифесте, который описывается кортежем

$$S_i = \langle v, a_i, F_i, p_i \rangle, \qquad (5)$$

где v обозначает версию продукта, a_i адаптер носителя с его версией, F_i множество автономных пакетов с версиями, p_i версию проектного расширения. Каждый компонент x указывает допустимые версии продукта: от v_x^{min} включительно до v_x^{max} исключительно. Поставка совместима, если

$$\forall x \in F_i \cup \{a_i, p_i\}: v_x^{min} \leq v < v_x^{max}, \qquad (6)$$

и регрессионная проверка этого состава прошла успешно. Верхней границей обычно служит следующая мажорная версия продукта, так как несовместимые изменения допускаются только в ней.

Выпуск версии проходит через автоматизированный конвейер: статический анализ, сборка расширений, развёртывание чистой тестовой базы, регрессионные тесты и публикация артефактов с привязкой к коммиту. Задача прослеживается от классификации до конкретной поставки, поэтому для любого заказчика можно восстановить точный состав установленного решения. Для каждого объекта метаданных дополнительно формируется машиночитаемая карточка, привязанная к версии исходного кода.

Экономическая оценка модульной модели сопровождения

Главный экономический эффект модульной модели связан с затратами на распространение изменений общей логики между внедрениями [9; 11]. Пусть за период планирования (например, месяц) выполняется m изменений общей логики, решение работает у N заказчиков, а час работы разработчика стоит c.

При модели клиентских копий каждое изменение один раз разрабатывается с трудоёмкостью t_d, а затем переносится в остальные $N - 1$ копий, и на перенос и проверку каждой уходит t_p. Затраты периода составят

$$Z_1 = c\, m\big(t_d + (N-1)\, t_p\big). \qquad (7)$$

В модульной модели изменение выполняется один раз, а заказчики получают его в составе очередного выпуска. Если за период выходит r выпусков, а обновление одного заказчика требует t_u, то

$$Z_2 = c\,(m\, t_d + r N t_u) + Z_I, \qquad (8)$$

где Z_I обозначает затраты периода на поддержку инструментария (конвейер сборки, репозитории, тестовые контуры). Экономия за период равна

$$\Delta Z = c\,\big(m(N-1)\, t_p - r N t_u\big) - Z_I. \qquad (9)$$

Модульная модель выгоднее, когда $\Delta Z > 0$. При $m t_p > r t_u$ отсюда получается пороговое число внедрений:

$$N^* = \frac{c\, m\, t_p + Z_I}{c\,(m\, t_p - r\, t_u)}. \qquad (10)$$

Условие $m t_p > r t_u$ означает, что изменения накапливаются в выпусках, и обновить заказчика дешевле, чем переносить к нему каждое изменение по отдельности. Единовременные затраты перехода K_0 (перенос решения в 1С:EDT, разделение на компоненты, настройка конвейера) окупаются за

$$T = K_0 / \Delta Z \qquad (11)$$

периодов. На практике t_p со временем растёт, потому что копии расходятся всё сильнее, так что реальная экономия только увеличивается. Кроме того, дешевле становится подключение нового заказчика и снижается риск ошибок расчёта, которые напрямую влияют на выручку.

Для иллюстрации возьмём условные значения параметров: $m = 20$ изменений в месяц, $t_p = 4$ ч, $t_u = 16$ ч, $r = 1$ выпуск в месяц, $N = 4$ заказчика, поддержка инструментария занимает 40 ч в месяц. Получаем $N^* = (80 + 40)/(80 - 16) \approx 1{,}9$. Значит, модульная модель выгодна уже со второго внедрения, а при четырёх внедрениях экономит $240 - 64 - 40 = 136$ ч в месяц, что составляет около 0,8 месячной нормы рабочего времени одного разработчика.

Апробация модели

Модель применялась при подготовке перехода действующей системы биллинга операторов морских терминалов на 1С:EDT и Git: по правилу (2) классифицированы все 98 объектов метаданных решения (табл. 1). Кандидаты в ядро относятся к пакетам «Контейнеры» и «Предварительный расчёт», перевод которых в ядро возможен отдельным продуктовым решением.

Таблица 1
Распределение объектов метаданных по классам компонентов

Вид объектов	Всего	Ядро	Пакеты	Кандидаты в ядро	Адаптер
Справочники	28	12	5	4	7
Планы видов характеристик	1	1	0	0	0
Документы	40	11	17	10	2
Регистры сведений	12	10	1	1	0
Регистры накопления	7	4	0	3	0
Перечисления	6	4	0	0	2
Отчёты и обработки	4	4	0	0	0
Итого	**98**	**46**	**23**	**18**	**11**
Доля, %	100	46,9	23,5	18,4	11,2

Источник: составлено автором по данным инвентаризации объектов конфигурации.

Почти половина объектов (46,9 %) относится к ядру, значит, ядро можно выпускать как законченный продукт. Пакеты вместе с кандидатами в ядро составляют 41,8 %, и решение о переводе кандидата в ядро разумно принимать по доле внедрений, в которых сценарий стал обязательным. Зависимость от учётной конфигурации сосредоточена в 11,2 % объектов: это справочники носителя (контрагенты, организации, валюты, ставки НДС, единицы измерения) и договорные документы. При такой доле подключение ядра к другим учётным конфигурациям через новые адаптеры выглядит вполне реальным. Объектов класса P в инвентаре продукта нет, клиентская специфика выносится в проектные расширения.

Хороший пример дают восемь регистров с историческим префиксом Bulk («навал»). По названию они относятся к навалочным грузам, а фактически служат общими механизмами периодического расчёта, учёта операций и контроля квот, поэтому отнесены к ядру. Классифицировать объект нужно по его бизнес-роли, а не по имени. Физические границы расширений предстоит окончательно проверить по графу зависимостей в 1С:EDT с помощью условий (3) и (4).

На основе модели подготовлен архитектурный стандарт решения: структура продуктового и клиентских репозиториев, модель ветвления, правила версионирования, формат манифеста поставки и порядок классификации новых требований до начала их реализации.

Заключение

Выручка инфраструктурного предприятия складывается из начислений за множество тарифицируемых операций, поэтому качество биллинга прямо влияет на экономические результаты. Главный риск при тиражировании биллинга на платформе «1С:Предприятие» связан с клиентскими копиями решения, из-за которых затраты на сопровождение растут вместе с числом внедрений.

В статье предложена продуктовая модель из ядра, функциональных пакетов, адаптеров конфигурации-носителя и проектных расширений, формализованы правило классификации функциональности и ограничения на граф зависимостей, описан инструментарий хранения, версионирования и поставки на основе 1С:EDT и Git. Экономическая модель показывает, с какого числа внедрений модульное сопровождение выгоднее копирования и за какой срок окупается переход. Апробация на 98 объектах действующей конфигурации подтвердила реализуемость модели: около половины объектов относится к ядру, а зависимость от учётной конфигурации затрагивает примерно 11 % объектов.

Следующим шагом исследования станет модель тарификации услуг инфраструктурного предприятия, реализуемая в ядре биллинговой системы.

Литература

1. Федеральный закон от 17.08.1995 № 147-ФЗ «О естественных монополиях» // КонсультантПлюс. URL: https://www.consultant.ru/document/cons_doc_LAW_7578/ (дата обращения: 28.09.2026).
2. Федеральный закон от 08.11.2007 № 261-ФЗ «О морских портах в Российской Федерации и о внесении изменений в отдельные законодательные акты Российской Федерации» // Президент России. URL: http://www.kremlin.ru/acts/bank/26440 (дата обращения: 28.09.2026).
3. Приказ Мининформсвязи России от 02.07.2007 № 73 «Об утверждении Правил применения автоматизированных систем расчетов» // Минцифры России. URL: https://digital.gov.ru/ru/documents/3918/ (дата обращения: 28.09.2026).
4. 1С:Enterprise Development Tools. Проект расширения конфигурации // ИТС 1С. URL: https://its.1c.ru/db/content/edtdoc/src/topics/t000538.html (дата обращения: 28.09.2026).
5. Лупанов Н. В. Некоторые аспекты создания инструментария цифровой трансформации на инфраструктурных предприятиях на базе 1С: EHS КОРП // Инжиниринг предприятий и управление знаниями (ИП&УЗ-2025): сб. науч. тр. XXVIII Рос. науч. конф. (Москва, 4–5 декабря 2025 г.). Т. 2. М.: РЭУ им. Г. В. Плеханова, 2026. С. 289–294.
6. Лупанов Н. В. Некоторые вопросы поддержки принятия решений в инфраструктурном предприятии // Инновации и инвестиции. 2026. № 3. С. 263–267.
7. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2018. 352 с.
8. Палагин Ю. И. Имитационное моделирование автоматизированных контейнерных терминалов морских портов // Транспорт: наука, техника, управление. 2025. № 8. С. 51–56. DOI: 10.36535/0236-1914-2025-08-7.
9. Скрипкин К. Г. Экономическая эффективность информационных систем. М.: ДМК Пресс, 2002. 256 с.
10. Уринцов А. И. Многоуровневые экономические информационные системы. 2003. URL: https://elibrary.ru/item.asp?id=20144163 (дата обращения: 28.09.2026).
11. Уринцов А. И., Дик В. В. Системы поддержки принятия решений и управления эффективностью бизнеса. 2009. URL: https://elibrary.ru/item.asp?id=20144153 (дата обращения: 28.09.2026).
12. Хрусталева Е. Расширения конфигураций. Адаптация прикладных решений с сохранением поддержки в облаках и на земле. Разработка в системе «1С:Предприятие 8.3». Litres, 2022. URL: https://books.google.com/books?id=usmJDwAAQBAJ (дата обращения: 28.09.2026).
13. Шепелин Г. И., Чижанькова И. В., Ниязбекова Ш. У. Проблематика цифровой и интеллектуальной трансформации в деятельности российских морских портов // Управленческий учет. 2023. № 6. С. 320–325. DOI: 10.25806/uu62023320-325.
14. Clements P., Northrop L. Software Product Lines: Practices and Patterns. Boston: Addison-Wesley, 2002. 563 p.
15. Driessen V. A successful Git branching model. 2010. URL: https://nvie.com/posts/a-successful-git-branching-model/ (дата обращения: 28.09.2026).
16. Kruchten P., Nord R. L., Ozkaya I. Technical Debt: From Metaphor to Theory and Practice // IEEE Software. 2012. Vol. 29, no. 6. P. 18–21. DOI: 10.1109/MS.2012.167.
17. Parnas D. L. On the Criteria to Be Used in Decomposing Systems into Modules // Communications of the ACM. 1972. Vol. 15, no. 12. P. 1053–1058. DOI: 10.1145/361598.361623.
18. Pohl K., Böckle G., van der Linden F. Software Product Line Engineering: Foundations, Principles and Techniques. Berlin; Heidelberg: Springer, 2005. 467 p. DOI: 10.1007/3-540-28901-1.
19. Preston-Werner T. Semantic Versioning 2.0.0. URL: https://semver.org/ (дата обращения: 28.09.2026).

Some aspects of creating billing tools at infrastructure enterprises based on 1c:enterprise

Lupanov N.V.
Plekhanov Russian University of Economics

JEL classification: C10, C50, C60, C61, C80, C87, C90

The article deals with creating replicable billing tools for infrastructure enterprises. Billing is understood as automated calculation of the cost of rendered services, and sea terminal operators are used as an example. When the same billing system based on the 1C:Enterprise platform is implemented for several customers, customer-specific modifications gradually become mixed with the common calculation logic. As a result, each customer ends up with its own copy of the solution, and maintenance costs grow with the number of implementations. The purpose of the study is to develop a model of modular construction and maintenance of a billing system that allows the solution to be replicated without customer copies of the core. The paper proposes a product model with four classes of components: the core, functional packages, carrier configuration adapters and project extensions. A rule for assigning functionality to these classes and constraints on the dependency graph are formalized. Tools for storing, versioning and delivering the solution based on 1C:EDT and Git are described, including a compatibility model for the delivery composition. An economic model of change propagation costs gives the number of implementations beyond which modular maintenance is more cost-effective than copying. The model was tested on 98 metadata objects of an operating configuration.

Keywords: billing, service pricing, infrastructure enterprise, sea terminal operator, instrumental methods of economics, modular architecture, 1C:Enterprise, 1C:EDT, version control, economic efficiency of information systems.

References

1. Federal Law of 17.08.1995 No. 147-FZ “On Natural Monopolies”. ConsultantPlus. (In Russ.). Available at: https://www.consultant.ru/document/cons_doc_LAW_7578/ (accessed 28.09.2026).
2. Federal Law of 08.11.2007 No. 261-FZ “On Sea Ports in the Russian Federation and on Amendments to Certain Legislative Acts of the Russian Federation”. (In Russ.). Available at: http://www.kremlin.ru/acts/bank/26440 (accessed 28.09.2026).
3. Order of the Ministry of Information Technologies and Communications of the Russian Federation of 02.07.2007 No. 73 “On Approval of the Rules for the Use of Automated Settlement Systems”. (In Russ.). Available at: https://digital.gov.ru/ru/documents/3918/ (accessed 28.09.2026).

4. 1C:Enterprise Development Tools. Configuration extension project. ITS 1C. (In Russ.). Available at: https://its.1c.ru/db/content/edtdoc/src/topics/t000538.html (accessed 28.09.2026).
5. Lupanov N. V. Some aspects of creating digital transformation tools at infrastructure enterprises based on 1C:EHS CORP. Inzhiniring predpriyatiy i upravlenie znaniyami (IP&UZ-2025): sbornik nauchnykh trudov XXVIII Rossiyskoy nauchnoy konferentsii [Enterprise Engineering and Knowledge Management (EE&KM-2025): proceedings of the 28th Russian Scientific Conference], Moscow, December 4–5, 2025. Vol. 2. Moscow: Plekhanov Russian University of Economics Publ., 2026, pp. 289–294. (In Russ.).
6. Lupanov N. V. Some issues of decision support in an infrastructure enterprise. Innovatsii i investitsii [Innovation and Investment], 2026, no. 3, pp. 263–267. (In Russ.).
7. Martin R. Chistaya arkhitektura. Iskusstvo razrabotki programmnogo obespecheniya [Clean Architecture: A Craftsman's Guide to Software Structure and Design]. Saint Petersburg: Piter Publ., 2018. 352 p. (In Russ.).
8. Palagin Yu. I. Simulation modeling of automated container terminals of seaports. Transport: nauka, tekhnika, upravlenie, 2025, no. 8, pp. 51–56. DOI: 10.36535/0236-1914-2025-08-7. (In Russ.).
9. Skripkin K. G. Ekonomicheskaya effektivnost' informatsionnykh sistem [Economic efficiency of information systems]. Moscow: DMK Press Publ., 2002. 256 p. (In Russ.).
10. Urintsov A. I. Mnogourovnevye ekonomicheskie informatsionnye sistemy [Multilevel economic information systems], 2003. (In Russ.). Available at: https://elibrary.ru/item.asp?id=20144163 (accessed 28.09.2026).
11. Urintsov A. I., Dik V. V. Sistemy podderzhki prinyatiya resheniy i upravleniya effektivnost'yu biznesa [Decision support systems and business performance management], 2009. (In Russ.). Available at: https://elibrary.ru/item.asp?id=20144153 (accessed 28.09.2026).
12. Khrustaleva E. Rasshireniya konfiguratsiy. Adaptatsiya prikladnykh resheniy s sokhraneniem podderzhki v oblakakh i na zemle. Razrabotka v sisteme «1S:Predpriyatie 8.3» [Configuration extensions. Adaptation of applied solutions while maintaining support in the cloud and on-premise. Development in 1C:Enterprise 8.3]. Litres Publ., 2022. (In Russ.). Available at: https://books.google.com/books?id=usmJDwAAQBAJ (accessed 28.09.2026).
13. Shepelin G. I., Chizhan'kova I. V., Niyazbekova Sh. U. Problems of digital and intellectual transformation in the activities of Russian seaports. Upravlencheskiy uchet, 2023, no. 6, pp. 320–325. DOI: 10.25806/uu62023320-325. (In Russ.).
14. Clements P., Northrop L. Software Product Lines: Practices and Patterns. Boston: Addison-Wesley, 2002. 563 p.
15. Driessen V. A successful Git branching model, 2010. Available at: https://nvie.com/posts/a-successful-git-branching-model/ (accessed 28.09.2026).
16. Kruchten P., Nord R. L., Ozkaya I. Technical debt: from metaphor to theory and practice. IEEE Software, 2012, vol. 29, no. 6, pp. 18–21. DOI: 10.1109/MS.2012.167.
17. Parnas D. L. On the criteria to be used in decomposing systems into modules. Communications of the ACM, 1972, vol. 15, no. 12, pp. 1053–1058. DOI: 10.1145/361598.361623.
18. Pohl K., Böckle G., van der Linden F. Software Product Line Engineering: Foundations, Principles and Techniques. Berlin; Heidelberg: Springer, 2005. 467 p. DOI: 10.1007/3-540-28901-1.
19. Preston-Werner T. Semantic Versioning 2.0.0. Available at: https://semver.org/ (accessed 28.09.2026).