

**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**Ярославский государственный университет им. П.Г.Демидова**

**УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ**  
**ПО ДИСЦИПЛИНЕ**

*ГИБКАЯ МЕТОДОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО  
ОБЕСПЕЧЕНИЯ*

Направление подготовки (специальность):  
02.04.02 Фундаментальная информатика и информационные технологии

Образовательная программа  
Искусственный интеллект и компьютерные науки

**очная форма обучения**

Составитель:  
**ПАРАМОНОВ ИЛЬЯ ВЯЧЕСЛАВОВИЧ,**  
**К.Ф.-М.Н., ДОЦЕНТ Ф-ТА ИВТ ЯРГУ ИМ. П.Г. ДЕМИДОВА**

## **Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)**

.

Основная литература:

- 1 Лауферман О.В., Лыгина Н.И. Разработка программного продукта: профессиональные стандарты, жизненный цикл, командная работа. Новосибирск: изд. НГТУ, 2019. 75 с. (ЭБС IPR BOOKS).
- 2 Зубкова, Т. М. Технология разработки программного обеспечения : учебное пособие / Т. М. Зубкова. — Санкт-Петербург : Лань, 2019. — 324 с. — ISBN 978-5-8114-3842-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/122176> (дата обращения: 05.10.2021). — Режим доступа: для авториз. пользователей.
3. Маран, М. М. Программная инженерия : учебное пособие / М. М. Маран. — Санкт-Петербург : Лань, 2021. — 196 с. — ISBN 978-5-8114-3032-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/169168> (дата обращения: 05.10.2021). — Режим доступа: для авториз. пользователей.

Дополнительная литература:

## **Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине (модулю)**

- 1 Электронный университет Moodle ЯрГУ URL: <https://moodle.uniyar.ac.ru/>
- 2 Единое окно доступа к образовательным ресурсам URL: <http://window.edu.ru/>

## **Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля), включая перечень информационных справочных систем (при необходимости)**

1. Электронная библиотека учебных материалов ЯрГУ ([http://www.lib.uniyar.ac.ru/opac/bk\\_cat\\_find.php](http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php)).
2. Информационная система "Единое окно доступа к образовательным ресурсам" (<http://www.edu.ru> (раздел Учебно-методическая библиотека) или по прямой ссылке <http://window.edu.ru/library>).
3. Электронно-библиотечная система «Университетская библиотека online» ([www.biblioclub.ru](http://www.biblioclub.ru)).

## **Перечень информационных технологий, используемых при изучении дисциплины, включая программное обеспечение**

В процессе осуществления образовательного процесса используются:

- для формирования текстов материалов для промежуточной и текущей аттестации – программы Microsoft Office,
- для поиска учебной литературы библиотеки ЯрГУ – Автоматизированная библиотечная информационная система "БУКИ-NEXT" (АБИС "Буки-Next").

**Учебно-методические указания и рекомендации  
к изучению тем лекционных и практических занятий, самостоятельной  
работе студентов**

Содержание дисциплины

Наименование раздела дисциплины	Название темы с кратким содержанием
1	<b>Понятие и основные положения гибкой методологии разработки.</b> Характерные черты гибкой методологии разработки. Существующие методики в рамках гибкой методологии. Создание собственных методик.
2	<b>Фундаментальные принципы гибкой разработки:</b> Работайте на результат (Work for outcome); Непродуманные исправления заводят в тупик (Quick fixes become quicksand); Критикуйте идеи, а не людей (Criticize ideas, not people); Будьте смелы в трудных ситуациях (Damn the torpedoes, go ahead).
3	<b>Принципы гибкого профессионального развития:</b> Будьте готовы к изменениям (Keep up with change); Инвестируйте в свою команду (Invest in your team); Умейте разучиваться (Know when to unlearn); Спрашивайте, пока не поймёте (Question until understand); Чувствуйте ритм (Feel the rhythm)
4	<b>Принципы гибкого проектирования и гибкой доставки разрабатываемого продукта пользователю:</b> Позвольте заказчику принимать решения (Let customers make decisions); Архитектура должна направлять, а не диктовать (Let design guide, not dictate); Обоснуйте использование выбранных технологий (Justify technology use); Продукт должен быть готов к выпуску в любой момент (Keep it releasable); Выполняйте раннюю и регулярную интеграцию (Integrate early, integrate often); Автоматизируйте развёртывание возможно раньше (Automate deployment early); Получайте обратную связь на основе демонстраций (Get frequent feedback using demos); Используйте короткие итерации, выпускайте приращениями (Use short iterations, release in increments); Контракты с фиксированной стоимостью — источник разочарований пользователей (Fixed prices are broken promises)

Наименование раздела	Название темы с кратким содержанием
5	<b>Принципы гибкой обратной связи:</b> Пишите тесты (Put angels on your shoulders); Практикуйте разработку, управляемую тестированием (Use it before you build it); Принимайте во внимание отличия в среде выполнения (Different makes a difference); Автоматизируйте приёмочное тестирование (Automate acceptance testing); Используйте правильные метрики (Measure real progress); Прислушивайтесь к пользователям (Listen to users)
6	<b>Принципы гибкого кодирования:</b> Программируйте осознанно и выразительно; Взаимодействуйте с коллегами посредством кода (Communicate in code); Осознанно ищите компромиссы (Actively evaluate trade-offs); Пишите код короткими сеансами (Code in increments); Предпочитайте простые решения (Keep it simple); Пишите связный код (Write cohesive code); Предпочитайте методы-команды методам-запросам (Tell, don't ask); Не нарушайте семантику интерфейса при наследовании (Substitute by contract)
7	<b>Принципы гибкой отладки приложений:</b> Записывайте принимаемые решения (Keep a solutions log); Предупреждения компилятора указывают на ошибки (Warning are really errors); Изолируйте проблему перед решением (Attack problems in isolation); Оповещайте об исключительных ситуациях (Report all exceptions); Предоставляйте содержательные сообщения об ошибках (Provide useful error messages)
8	<b>Принципы гибкого взаимодействия внутри команды:</b> Выделите время для регулярных совместных обсуждений проекта (Schedule regular face time); Архитекторы должны писать код (Architects must write code); Практикуйте коллективное владение кодом (Practice collective ownership); Будьте наставником (Be a mentor); Давайте ключ к решению, а не само решение (Allow people to figure it out); Публикуйте только готовый код (Share code only when ready); Рецензируйте код (Review code); Держите коллег и заказчика в курсе дел (Keep others informed)

### Задания для проведения семинарских занятий:

#### Тема 1 «Понятие и основные положения гибкой методологии разработки».

1. Оцените возможности применения гибкой методологии в рамках каскадной и итеративной моделей процесса разработки?
2. Объясните, каким образом идеи манифеста гибкой разработки нашли своё выражение в конкретных методиках.
3. Выделите основные риски процесса разработки, на управление которыми направлена гибкая методология и конкретные методики в рамках гибкой методологии.

4. Почему попытки одномоментного внедрения гибкой методологии в процесс разработки практически всегда заканчиваются неудачей?

## Тема 2 «Фундаментальные принципы гибкой разработки».

1. Почему принцип «работайте на результат» столь критичен для команд разработчиков, придерживающихся методологии гибкой разработки, но может уходить на второй план в командах, придерживающихся традиционной методологии? К каким последствиям может приводить нарушение этого принципа в тех и других командах?

2. Объясните, почему механическое выполнение требований начальника может быть пагубным для процесса разработки. Какие требования предъявляет принцип «работайте на результат» менеджеру проекта, реализуемого в рамках гибкой методологии?

3. Вспомните из собственной практики случай, когда Вы пренебрегли принципом «непродуманные исправления заводят в тупик» и это повлекло негативные последствия. Предложите шаги, которые надо было предпринять для правильного решения проблемы.

4. Предложите способы борьбы с внесением разработчиками непродуманных изменений.

5. Объясните, в чём заключаются риски сопровождения продукта, обусловленные непродуманными исправлениями.

6. Найдите взаимосвязь между принципами «работайте на результат» и «критикуйте идеи, а не людей». Как эта взаимосвязь может оказаться полезной при внедрении гибкой методологии в процесс разработки?

7. Представьте, что при разработке некоторого программного продукта обнаружен серьёзный архитектурный дефект, который существенно мешает решению текущих задач разработчиков; при этом исправление данного дефекта потребует столь больших временных затрат, что сроки сдачи продукта в эксплуатацию будут сорваны. Предложите возможные варианты решения данной проблемы.

## Тема 3 «Принципы гибкого профессионального развития».

1. Проанализируйте причины устаревания знаний и навыков. Приведите примеры быстро и медленно устаревающих знаний. Какие категории знаний делают программиста конкурентоспособным?

2. Следует ли разработчикам быть компетентными в технологиях, применяемых дизайнерами, менеджерами, тестировщиками?

3. Проанализируйте, как принцип «инвестируйте в свою команду» может повлиять на выбор технологий для будущего проекта.

4. Достаточно ли разработчику участия во внутренних семинарах компании для того, чтобы оставаться конкурентоспособным на рынке труда? Приведите примеры.

5. Применим ли шаблон программирования «модель-вид-контроллер» для создания консольных приложений?

6. Изучите техники управления рабочим временем «Pomodoro technique» и «Getting things done». Оцените их применимость в рамках гибкой методологии разработки.

## Тема 4 «Принципы гибкого проектирования и гибкой доставки разрабатываемого продукта пользователю».

1. По каким признакам разработчик должен определить, что для принятия решения по некоторому вопросу он должен обратиться к заказчику? Какими навыками должен обладать разработчик для того, чтобы эффективно применять принцип «позвольте заказчику принимать решения»?

2. Почему в большинстве случаев желательно, чтобы разработчик не только делегировал заказчику принятие бизнес-решения, но также предлагал различные варианты таких решений?
3. Какие виды архитектурных артефактов и какие инструменты могут быть полезны для разработки в рамках гибкой методологии?
4. Определите критерии, которыми следует руководствоваться при выборе технологий и инструментов для проектов разработки ПО разных типов (например, мобильные приложения, веб-приложения, информационные системы уровня предприятия, системы реального времени и т. д.).
5. Рассмотрите несколько современных языков программирования, фреймворков, технологий и сформулируйте условия их применимости в разработке программного обеспечения.
6. Раскройте типичные причины возникновения неожиданных проблем при выполнении системной интеграции.
7. Ознакомьтесь с возможностями, предоставляемыми серверами непрерывной интеграции. Каким образом такие серверы могут быть использованы для воплощения в жизнь принципа «выполняйте раннюю и регулярную интеграцию»?
8. Подготовьте детальный обзор инструментов, которые могут применяться для автоматизации развёртывания, а также их возможностей.
9. Какие требования предъявляются к промежуточной версии продукта, показываемой во время демонстраций для получения обратной связи?
10. Представляют ли изменения, выявляемые в результате каждой демонстрации, угрозу для первоначальных планов доставки функциональности в результате приращений?

#### Тема 5 «Принципы гибкой обратной связи».

1. Почему неавтоматические тесты не позволяют следовать принципу «пишите тесты» в полном объёме?
2. Идентифицируйте все возможные роли, которые могут играть автоматические тесты в проекте.
3. Исследуйте, в каких случаях рационально применять разработку, управляемую тестированием, а в каких — писать тесты после написания кода.
4. Каким образом применение техники TDD может оказать влияние на архитектуру разрабатываемой системы? Упрощает ли разработка, управляемая тестированием, выработку архитектуры системы или усложняет её?
5. Для разных типов приложений (настольные, мобильные, веб-приложения и т. д.) определите возможные отличия сред выполнения, которые надо принимать во внимание во время разработки.
6. Ознакомьтесь со специализированными инструментами автоматизации приёмочного тестирования. В каких случаях их рационально применять?
7. Сравните бэклог с планом реализации проекта, разрабатываемого в рамках каскадной модели. В чём сходства и различия между этими двумя инструментами?
8. Как, используя диаграмму сгорания задач, можно строить планы и прогнозы относительно проекта?
9. Какие риски могут оказывать существенное влияние на время решения задач разработки? Предложите меры по управлению этими рисками.

10. Вспомните «типичную» ошибку, которую Вы регулярно совершаете при работе с каким-либо приложением. Обусловлена ли она ошибкой разработчика? Можете ли Вы предложить лучшее решение?

11. Тесное взаимодействие с заказчиком может провоцировать его изменять требования к разрабатываемому продукту слишком часто. Предложите способы решения данной проблемы.

#### Тема 6 «Принципы гибкого кодирования».

1. Выберите и сравните несколько языков программирования на предмет их внутренней выразительности. Оцените, насколько существенным может быть влияние выбора языка программирования на удобочитаемость и сложность программного кода.

2. Рассмотрите 10-15 приёмов повышения удобочитаемости и выразительности кода, описанных в литературе. Определите, как конкретно влияет применение данных приёмов на удобочитаемость и выразительность.

3. Найдите фрагмент собственного кода, непонятного или с трудом понятного без дополнительной документации. С помощью серии рефакторингов сделайте этот код самодокументированным.

4. Какие качества наиболее важны для продуктов следующих типов: настольное приложение, мобильное приложение, веб-приложение, система реального времени, корпоративная информационная система?

5. Известно, что для решения многих задач разработки необходимо «погружение» в предметную область и детали реализации системы. Не вступает ли необходимость такого «погружения» в противоречие с принципом «пишите код короткими сеансами»?

6. Помимо переусложнённых решений задач разработки встречаются также переупрощённые. Исследуйте, где проходит грань между простыми решениями и переупрощёнными. Приведите примеры.

7. Каким образом метод карт CRC помогает следовать принципу «пишите связный код»?

8. Рассмотрите преимущества и недостатки методов-команд, имеющих возвращаемые значения. Приведите примеры.

9. Приведите примеры нарушения семантики интерфейсов при наследовании, используя примеры из собственного кода и стандартных библиотек. Покажите, к каким конкретным патологиям в коде приводят данные нарушения.

#### Тема 7 «Принципы гибкой отладки приложений».

1. Рассмотрите различные способы организации базы знаний для хранения информации о принимаемых решениях, а также инструменты, упрощающие такую организацию. Для чего может потребоваться выполнение сопровождения базы знаний и как его осуществлять?

2. Отберите 5-10 распространённых предупреждений компиляторов различных языков программирования и объясните, о каких возможных проблемах в коде эти предупреждения сигнализируют.

3. Вспомните из собственной практики случай, когда Вы пренебрегли принципом «изолируйте проблему перед решением», и это повлекло негативные последствия. Предложите шаги, которые надо было предпринять для правильного решения проблемы.

4. Предложите варианты архитектуры, поддерживающие оповещение пользователя об исключительных ситуациях, для различных типов приложений (консольное, настольное приложение с графическим интерфейсом, мобильное, веб-приложение).

5. Какую роль могут играть системные журналы (log) на различных этапах процесса разработки (кодирование, тестирование, отладка, сопровождение)? Для каких целей следует использовать журналы, а для каких нет? Какую информацию имеет смысл выводить в журнал?

#### Тема 8 «Принципы гибкого взаимодействия внутри команды».

1. В чём состоит специфика организации взаимодействия в распределённых командах разработчиков по сравнению с командами, сосредоточенными в одном пространстве? Предложите эффективные способы взаимодействия для распределённых команд.
2. Каким требованиям должна удовлетворять первоначальная, выработанная до реализации проекта архитектура, чтобы обеспечить выполнение принципа «Архитекторы должны писать код»? Какие меры должны предпринимать разработчики, изменяя архитектуру проекта во время его разработки, чтобы обеспечить его сопровождаемость?
3. Если команда следует принципу коллективного владения кодом, то для её проектов могут представлять некоторую опасность начинающие и неопытные разработчики, а также разработчики, склонные вносить быстрые непродуманные изменения. В чём могут заключаться эти опасности и как с ними бороться?
4. Выполняя функции наставника, разработчик вынужден отвлекаться от решения собственных задач, что приводит к дополнительным незапланированным временным затратам. Как это может сказываться на общем времени реализации проекта? Рассмотрите различные ситуации.
5. Занимаясь наставничеством, разработчик делится своими знаниями и опытом с другими членами команды, до некоторой степени теряя при этом свою исключительность в команде. Не несёт ли это угрозы для его карьеры?
6. Как принцип «Давайте ключ к решению, а не само решение» позволяет совершенствовать управление внутри команды разработчиков?
7. Раскройте связь между принципом «Публикуйте только готовый код» и другими принципами гибкой разработки.
8. Как механизм ветвей (branches) систем управления версиями помогает следовать принципу «Публикуйте только готовый код»?
9. Сравните автоматическое тестирование, статический анализ и рецензирование кода по возможностям обнаружения ошибок различных типов.
10. Многие разработчики боятся держать своего начальника в курсе возникающих проблем, опасаясь обвинений в некомпетентности. Проанализируйте данную проблему и предложите решения.

#### Критерии оценки

«Отлично» – ответ на вопросы показывает всестороннее знание темы, изученной литературы, изложен логично, аргументировано и в полном объеме. Основные понятия, выводы и обобщения сформулированы убедительно и доказательно. Продемонстрированы широкие знания в области гибкой методологии разработки программного обеспечения и их глубокое понимание и умение делать выводы.

«Хорошо» – ответ на вопросы основан на твердом знании темы. Возможны недостатки в систематизации или в обобщении материала, неточности в выводах. Продемонстрированы хорошие знания в области гибкой методологии разработки программного обеспечения.

«Удовлетворительно» – ответ на вопросы базируется на знании основ предмета, но имеются значительные пробелы в изложении материала, затруднения в его изложении и систематизации, выводы слабо аргументированы, в содержании допущены теоретические

ошибки. Продемонстрированы элементарные знания в области гибкой методологии разработки программного обеспечения, но слабое умение делать выводы из них.

«Неудовлетворительно» – оценивается ответ на вопросы, в котором обнаружено неверное изложение темы, систематизации знаний, обобщений и выводов нет. Общее понимание гибкой методологии разработки программного обеспечения слабое, отрывочное или отсутствует.