МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ярославский государственный университет им. П.Г. Демидова

Кафедра компьютерных сетей

УТВЕРЖДАЮ
Декан факультета ИВТ
_____ Д.Ю. Чалый
«_24_» _мая__ 2022 г.

Рабочая программа дисциплины
«Гибкая методология разработки программного обеспечения»

Направление подготовки
01.04.02 Прикладная математика и информатика

Направленность (профиль)
«Математические основы искусственного интеллекта»

Квалификация выпускника
Магистр

Форма обучения
очная

Программа рассмотрена
на заседании кафедры
от «22» марта 2022 г.,
протокол № 7

Программа одобрена НМК
факультета ИВТ
протокол № 6 от
«18» апреля 2022 г. года

Ярославль

1. Цели освоения дисциплины

Целью дисциплины «Гибкая методология программного обеспечения» является изучение принципов гибкой методологии промышленной разработки программного обеспечения и особенностей управления разработкой ПО, основанной на этой методологии. Рассматриваются распространенные методики гибкой разработки программного обеспечения, основные ценности, принципы и нормы практики гибкой разработки, роли участников проекта, особенности планирования и проведения отдельных стадий процесса разработки программного обеспечения и подходы к эффективному применению методики на практике.

Для изучения данной дисциплины необходимо знакомство обучающихся с основными принципами промышленной разработки в рамках традиционной методологии, а также основных положений программной инженерии.

2. Место дисциплины в структуре ОП магистратуры

Дисциплина «Гибкая методология программного обеспечения» относится к вариативной части ОП магистратуры.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения ОП магистратуры

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции (код и формулировка)**	**Перечень планируемых результатов обучения**
Профессиональные компетенции		
ПК-3 Способен управлять проектами в области информационных технологий	ПК-3.2 Выбирает наиболее эффективные методы управления разработкой программных средств и проектов	Знать: технологическое развитие центров обработки данных, наращивание и поддержание технологических мощностей и компетенций подразделений Уметь: Участвовать в создании (модернизации) общедоступных платформ для хранения наборов данных, соответствующих методологиям описания, сбора и разметки данных; хранения наборов данных (в том числе звуковых, речевых, медицинских, метеорологических, промышленных данных и данных систем видеонаблюдения) на общедоступных платформах для обеспечения потребностей

		организаций- разработчиков в области искусственного интеллекта Владеть навыками: личное участие в проектах в роли архитектора центра обработки данных, технологического эксперта

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 2 зач.ед., 72 акад.час.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
			Контактная работа						
			лекции	практические	лабораторные	консультации	аттестационные испытания	самостоятельная работа	
1.	Понятие и основные положения гибкой методологии разработки.	2	1		1			4	
2	Фундаментальные принципы гибкой разработки:	2	2		2			4	
3	Принципы гибкого профессионального развития	2	1		1			4	
4	Принципы гибкого проектирования и гибкой доставки разрабатываемого продукта пользователю	2	2		2			4	
5	Принципы гибкой обратной связи	2	1		1			4	
6	Принципы гибкого	2	2		2			6	

	кодирования								
7	Принципы гибкой отладки приложений	2	1		1			5	
8	Принципы гибкого взаимодействия внутри команды	2	7		7			4,7	
	Всего		**17**		**17**			**35,7**	

Содержание разделов дисциплины:

Понятие и основные положения гибкой методологии разработки. Характерные черты гибкой методологии разработки. Существующие методики в рамках гибкой методологии. Создание собственных методик.

Фундаментальные принципы гибкой разработки: 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).

Принципы гибкого профессионального развития: 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)

Принципы гибкого проектирования и гибкой доставки разрабатываемого продукта пользователю: 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)

Принципы гибкой обратной связи: 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)

Принципы гибкого кодирования: 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)

Принципы гибкой отладки приложений: 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)

Принципы гибкого взаимодействия внутри команды: 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)

5. Образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция (или лекция общего курса) – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Требования к академической лекции: современный научный уровень и насыщенная информативность, убедительная аргументация, доступная и понятная речь, четкая структура и логика, наличие ярких примеров, научных доказательств, обоснований, фактов.

Лабораторное занятие – занятие, посвященное освоению и углублению конкретных умений и навыков и закреплению полученных на лекции знаний.

6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем (при необходимости)

В процессе осуществления образовательного процесса используются:

– для формирования текстов материалов для промежуточной и текущей аттестации, для разработки документов, презентаций, для работы с электронными таблицами - программы OfficeStd 2013 RUSOLPNLAcdmc 021-10232, LibreOffice (свободное), издательская система LaTeX;

- компиляторы с высокоуровневых языков программирования;

– для поиска учебной литературы библиотеки ЯрГУ– Автоматизированная библиотечная информационная система "БУКИ-NEXT" (АБИС "Буки-Next").

– PacketTracer 6.3, CiscoSDM, CiscoNetworkAssistant, CiscoConfigurationProfessional.

7. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

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.

8. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

- специальные помещения:

-учебные аудитории для проведения занятий лекционного типа и лабораторных занятий;

- учебные аудитории для проведения групповых и индивидуальных консультаций,

- учебные аудитории для проведения текущего контроля и промежуточной аттестации;

-помещения для самостоятельной работы;

-помещения для хранения и профилактического обслуживания технических средств обучения.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации.

Число посадочных мест в лекционной аудитории больше либо равно списочному составу потока, а в аудитории для лабораторных занятий – списочному составу группы обучающихся.

- фонд библиотеки.

Компьютерные классы, оборудованные ПЭВМ класса не ниже IntelCore2Duo , 4gbRAM, 60GHDDc установленным программным обеспечением: Windows7/8/10, Linux, PacketTracer 6.3, CiscoSDM, CiscoNetworkAssistant, CiscoConfigurationProfessional. Из расчета одна ПЭВМ на одного человека.

Автор(ы) : доц., к.ф.-м. н. И. В. Парамонов

Приложение №1 к рабочей программе дисциплины «Гибкая методология разработки программного обеспечения»

Фонд оценочных средств для проведения текущей и промежуточной аттестации студентов по дисциплине

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?

Основная литература:

3. Лауферман О.В., Лыгина Н.И. Разработка программного продукта: профессиональные стандарты, жизненный цикл, командная работа. Новосибирск: изд. НГТУ, 2019. 75 с. (ЭБС IPR BOOKS).

4. 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.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.

Приложение №2 к рабочей программе дисциплины
«Гибкая методология разработки программного обеспечения»

Методические указания для студентов по освоению дисциплины

Основной формой изложения учебного материала по дисциплине «Гибкая методология разработки программного обеспечения» являются практические занятия, на которых происходит закрепление материала путем применения его к конкретным задачам и отработка практических навыков.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются на лекциях и лабораторных занятиях, при необходимости по наиболее трудным темам проводятся дополнительные консультации. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной на консультациях, лабораторных занятиях или из учебной литературы.

Освоить вопросы, излагаемые в процессе изучения дисциплины «Гибкая методология разработки программного обеспечения» самостоятельно студенту крайне сложно. Это связано со сложностью изучаемого материала и большим объемом курса. Поэтому посещение всех аудиторных занятий является совершенно необходимым. Без упорных и регулярных занятий в течение семестра сдать зачет по итогам изучения дисциплины студенту практически невозможно.

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

Для самостоятельной работы особенно рекомендуется использовать учебную литературу.

Также для подбора учебной литературы рекомендуется использовать широкий спектр интернет-ресурсов:

1. Электронно-библиотечная система «Университетская библиотека online» (www.biblioclub.ru) - электронная библиотека, обеспечивающая доступ к наиболее востребованным материалам-первоисточникам, учебной, научной и художественной литературе ведущих издательств (*регистрация в электронной библиотеке – только в сети университета. После регистрации работа с системой возможна с любой точки доступа в Internet.).

2.Информационная система "Единое окно доступа к образовательным ресурсам" (http://window.edu.ru/library).

Целью создания информационной системы "Единое окно доступа к образовательным ресурсам" (ИС "Единое окно ") является обеспечение свободного доступа к интегральному каталогу образовательных интернет-ресурсов и к электронной библиотеке учебно-методических материалов для общего и профессионального образования.

Информационная система "Единое окно доступа к образовательным ресурсам" создана по заказу Федерального агентства по образованию в 2005-2008 гг. Головной разработчик проекта - Федеральное государственное автономное учреждение Государственный научно-исследовательский институт информационных технологий и телекоммуникаций (ФГАУ ГНИИ ИТТ "Информика") www.informika.ru.

Для самостоятельного подбора литературы в библиотеке ЯрГУ рекомендуется использовать:

1. Личный кабинет (http://lib.uniyar.ac.ru/opac/bk_login.php)дает возможность получения on-line доступа к списку выданной в автоматизированном режиме литературы, просмотра и копирования электронных версий изданий сотрудников университета (учеб. и

метод. пособия, тексты лекций и т.д.) Для работы в «Личном кабинете» необходимо зайти на сайт Научной библиотеки ЯрГУ с любой точки, имеющей доступ в Internet, в пункт меню «Электронный каталог»; пройти процедуру авторизации, выбрав вкладку «Авторизация», и заполнить представленные поля информации.

2. Электронная библиотека учебных материалов ЯрГУ

(http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php) содержит более 2500 полных текстов учебных и учебно-методических материалов по основным изучаемым дисциплинам, изданных в университете. Доступ в сети университета, либо по логину/паролю.

3. Электронная картотека «Книгообеспеченность»

(http://www.lib.uniyar.ac.ru/opac/bk_bookreq_find.php) раскрывает учебный фонд научной библиотеки ЯрГУ, предоставляет оперативную информацию о состоянии книгообеспеченности дисциплин основной и дополнительной литературой, а также цикла дисциплин и специальностей. Электронная картотека «Книгообеспеченность» доступна в сети университета и через Личный кабинет.