МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)

Физический факультет
Кафедра автоматизации физико-технических исследований

УТВЕРЖДАЮ
Декан ФФ, д.ф.-м.н
В.Е.Блинов
«__» _____ 2022 г.

Рабочая программа дисциплины

ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

направление подготовки: **03.04.02 Физика**
направленность (профиль): **Информационные процессы и системы**

Форма обучения
Очная

Семестр	Общий объем	Виды учебных занятий (в часах)				Промежуточная аттестация (в часах)				
		Контактная работа обучающихся с преподавателем			Самостоятельная работа, не включая период сессии	Самостоятельная подготовка к промежуточной аттестации	Контактная работа обучающихся с преподавателем			
		Лекции	Практические занятия	Лабораторные занятия			Консультации	Зачет	Дифференцированный зачет	Экзамен
1	2	3	4	5	6	7	8	9	10	11
1	108	32		48	26				2	
Всего 108 часов / 3 зачетных единицы, из них: - контактная работа 82 часа										
Компетенции ПК-2										

Руководитель программы
д.ф.-м.н. И. Б. Логашенко

Новосибирск, 2022

Содержание

1. Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы

Целями освоения дисциплины «Технология разработки программного обеспечения» являются:

- ознакомление с современными языками программирования, их классификацией и областями их применения;
- освоение различных методов абстрагирования, обеспечения модульности и других аспектов проектирования программных систем;
- повышение профессиональной эрудиции.

Дисциплина нацелена на формирование у обучающегося профессиональной компетенции:

Результаты освоения образовательной программы (компетенции)	Индикаторы	Результаты обучения по дисциплине
ПК-2 Способен использовать специализированные знания в области физики при постановке и решении задач в научно-исследовательской деятельности с помощью современной аппаратуры и информационно-телекоммуникационных технологий в соответствии с профилем подготовки в зависимости от специфики объекта исследования	**ПК -2.1.** Проводит научные изыскания в избранной области экспериментальных и/или теоретических физических исследований с помощью современной аппаратуры и информационно-телекоммуникационных технологий в соответствии с профилем подготовки в зависимости от специфики объекта исследования. **ПК -2.2.** Применяет теоретические основы и базовые представления научного исследования в выбранной области фундаментальной и/или экспериментальной физики в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Знать** методы проектирования программных средств вычислительной техники; методы и алгоритмы объектно-ориентированного, функционального, аспектно-ориентированного программирования **Уметь** комбинировать различные языки и системы программирования, а также методы проектирования с целью оптимального решения поставленных задач; Использовать типовые программные продукты, ориентированные на решение научных, проектных и технологических задач. **Владеть** рядом современных функциональных, динамических и аспектно-ориентированных языков, а также соответствующими им методами проектирования; навыками самостоятельной научно-исследовательской деятельности.

2. Место дисциплины в структуре образовательной программы

Курс относится к циклу профессиональных дисциплин и реализуется в осеннем семестре 1-го курса для магистрантов, обучающихся по направлению подготовки 03.04.02 Физика, направленность «Информационные процессы и системы». Для успешного освоения курса необходимо знание английского языка на уровне чтения документации, основ структурного и объектно-ориентированного программирования.

После изучения курса студенты могут продолжить изучать курсы, посвященные проектированию специализированных программно-аппаратных средств для задач высокопроизводительной обработки данных в темпе поступления, а также разработки автономных устройств обработки данных с батарейным питанием.

3. Трудоёмкость дисциплины в зачётных единицах с указанием количества академических часов, выделенных на контактную работу обучающегося с преподавателем (по видам учебных занятий) и на самостоятельную работу

Семестр	Общий объем	Виды учебных занятий (в часах)				Промежуточная аттестация (в часах)				
		Контактная работа обучающихся с преподавателем			Самостоятельная работа, не включая период сессии	Самостоятельная подготовка к промежуточной аттестации	Контактная работа обучающихся с преподавателем			
		Лекции	Практические занятия	Лабораторные занятия			Консультации	Зачет	Дифференцированный зачет	Экзамен
1	2	3	4	5	6	7	8	9	10	11
1	108	32		48	26				2	
Всего 108 часов / 3 зачетные единицы, из них: - контактная работа 82 часа										
Компетенции ПК-2										

Реализация дисциплины предусматривает практическую подготовку при проведении следующих видов занятий, предусматривающих участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью: лекции, лабораторные занятия, самостоятельная работа студента, дифференцированный зачет.

Программой дисциплины предусмотрены следующие виды контроля:

Текущий контроль: опрос студентов в начале каждого занятия, решение задач

Промежуточная аттестация: дифференцированный зачет

Общая трудоемкость рабочей программы дисциплины составляет **108** академических часа / **3** зачетные единицы:

- занятия лекционного типа – 32 часа;
- лабораторные занятия – 48 часов;
- самостоятельная работа обучающегося в течение семестра, не включая период сессии – 26 часов;
- промежуточная аттестация (дифференцированный зачет) – 2 часа.

Объём контактной работы обучающегося с преподавателем (занятия лекционного типа, лабораторные занятия, дифференцированный зачет) составляет 82 часа.

4. Содержание дисциплины, структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

Дисциплина «Технология разработки программного обеспечения» представляет собой полугодовой курс, читаемый на 1 курсе физического факультета НГУ в 1 семестре. Общая трудоемкость дисциплины «Технология разработки программного обеспечения» составляет **3** зачетные единицы / **108** академических часов.

№ п/п	Раздел дисциплины	Неделя семестра	Всего	Виды учебных занятий, включая самостоятельную работу студентов и трудоемкость (в часах)			Промежуточная аттестация (в период сессии) (в часах)
				Аудиторные часы		Сам. работа в течение семестра (не включая период сессии)	
				Лекции (кол-во часов)	Лабораторные занятия (кол-во часов)		
1	2	3		5	6	7	8
1	Общая классификация языков по назначению и модели исполнения.	1	5	2	3		
2	Язык Ruby: основные конструкции языка, коллекции.	2-4	17	6	9	2	
3	Особенности объектной модели Ruby.	5	7	2	3	2	
4	Регулярные выражения. Классы символов.	6	7	2	3	2	
5	Классификация языков по парадигмам программирования.	7	7	2	3	2	
6	Общие характеристики семейства языков Lisp	8	7	2	3	2	
7	Функциональные возможности Clojure	9	7	2	3	2	
8	Императивные возможности Clojure	10-11	14	4	6	4	
9	Управляемая кодогенерация. Макросы в Lisp	12	7	2	3	2	
10	Понятие о проблемно-специфичных языках (DSL) и языках сценариев.	13	7	2	3	2	
11	Динамические объектные модели. CLOS	14	7	2	3	2	
12	Сквозная функциональность (cross-cutting concerns)	15	7	2	3	2	
13	Аспектно-ориентированное программирование (АОП).	16	7	2	3	2	
	Дифференцированный зачет		2				2
Всего			**108**	**32**	**48**	**26**	**2**

Программа и основное содержание лекций (32 часа)

1. Общая классификация языков по назначению и модели исполнения. Общие свойства динамических языков (динамическая типизация, модель трансляции и исполнения) **(2 часа)**
2. Язык Ruby: основные конструкции языка, коллекции. Функциональный стиль программирования в Ruby: блоки и замыкания. Итераторы. Реорганизующее присваивание. **(6 часов)**
3. Особенности объектной модели Ruby: унифицированность объектного представления, модули и примеси, инкапсуляция. Динамическое изменение классов, элементы Meta-Object Protocol (MOP) в Ruby. JRuby и взаимодействие с Java, Java Scripting API. **(2 часа)**

4. Регулярные выражения. Классы символов. Жадные и нежадные выражения. Основные операции с регулярными выражениями. **(2 часа)**

5. Классификация языков по парадигмам программирования. Функциональное программирование (ФП). Неподвижное состояние объекта как ключевое отличие ФП от ООП. Чистые функции, функции высших порядков. Функции, как объекты первого класса. Лексические контексты, анонимные функции, замыкания. Основные семейства функциональных языков. Историческая связь динамических и функциональных языков. **(2 часа)**

6. Общие характеристики семейства языков Lisp: единое представление кода и данных, S-выражения, модель трансляции и исполнения, REPL. Язык Clojure, как современный представитель семейства Lisp: основные структуры языка. Компонентное тестирование в Clojure. **(2 часа)**

7. Функциональные возможности Clojure: коллекции, реорганизующее присваивание, мемоизация, отложенные вычисления, бесконечные структуры данных. Абстрагирование данных с помощью функциональных примитивов (пары, числа Черча). Моделирование времени с помощью потоков. Символьные вычисления. Преимущества и недостатки ФП в сравнении с ООП. **(2 часа)**

8. Императивные возможности Clojure. Software Transactional Memory. Многопоточность. Ссылки, атомы, агенты, переменные, виды транзакций. Взаимодействие с Java. **(4 часа)**

9. Управляемая кодогенерация. Макросы в Lisp (на примере Clojure). Модель исполнения макросов. Макросы, как способ расширения языка. **(2 часа)**

10. Понятие о проблемно-специфичных языках (DSL) и языках сценариев. Методы построения и генерации DSL. **(2 часа)**

11. Динамические объектные модели. CLOS: обобщенный динамический полиморфизм, обобщенные функции и мультиметоды, вспомогательные методы. Реализация элементов CLOS в Clojure. Интроспекция, введение в MOP. **(2 часа)**

12. Сквозная функциональность (cross-cutting concerns), проблема модульности. Традиционные методы обеспечения модульности в условиях сквозной функциональности. **(2 часа)**

13. Аспектно-ориентированное программирование (АОП). Динамические лексические контексты, их реализация в Clojure. Элементы АОП в CLOS. **(2 часа)**

Программа лабораторных занятий (48 часов)

8 часов	Разбор теоретической части, решение заданий по темам: 1.1. Общая классификация языков по назначению и модели исполнения. Общие свойства динамических языков (динамическая типизация, модель трансляции и исполнения) 1.2. Язык Ruby: основные конструкции языка, коллекции. Функциональный стиль программирования в Ruby: блоки и замыкания. Итераторы. Реорганизующее присваивание. 1.3. Особенности объектной модели Ruby: унифицированность объектного представления, модули и примеси, инкапсуляция. Динамическое изменение классов, элементы Meta-Object Protocol (MOP) в Ruby. JRuby и взаимодействие с Java, Java Scripting API.

	1.4. Регулярные выражения. Классы символов. Жадные и нежадные выражения. Основные операции с регулярными выражениями.
16 часов	Разбор теоретической части, решение заданий по темам 2.1. Классификация языков по парадигмам программирования. Функциональное программирование (ФП). Неподвижное состояние объекта как ключевое отличие ФП от ООП. Чистые функции, функции высших порядков. Функции, как объекты первого класса. Лексические контексты, анонимные функции, замыкания. Основные семейства функциональных языков. Историческая связь динамических и функциональных языков. 2.2. Общие характеристики семейства языков Lisp: единое представление кода и данных, S-выражения, модель трансляции и исполнения, REPL. Язык Clojure, как современный представитель семейства Lisp: основные структуры языка. Компонентное тестирование в Clojure. 2.3. Функциональные возможности Clojure: коллекции, реорганизующее присваивание, мемоизация, отложенные вычисления, бесконечные структуры данных. Абстрагирование данных с помощью функциональных примитивов. Моделирование времени с помощью потоков. Символьные вычисления. Преимущества и недостатки ФП в сравнении с ООП. 2.4. Императивные возможности Clojure. Software Transactional Memory. Многопоточность. Ссылки, атомы, агенты, переменные, виды транзакций. Взаимодействие с Java.
24 часа	Разбор теоретической части, решение заданий по темам 3.1. Управляемая кодогенерация. Макросы в Lisp (на примере Clojure). Модель исполнения макросов. Макросы, как способ расширения языка. 3.2. Понятие о проблемно-специфичных языках (DSL) и языках сценариев. Методы построения и генерации DSL. 3.3. Динамические объектные модели. CLOS: обобщенный динамический полиморфизм, обобщенные функции и мультиметоды, вспомогательные методы. Реализация элементов CLOS в Clojure. Интроспекция, введение в MOP. 3.4. Сквозная функциональность (cross-cutting concerns), проблема модульности. Традиционные методы обеспечения модульности в условиях сквозной функциональности. 3.5. Аспектно-ориентированное программирование (АОП). Динамические лексические контексты, их реализация в Clojure. Элементы АОП в CLOS, 3.6. Применение АОП в проектирование. Преимущества и недостатки по сравнению с традиционными методами проектирования. Примеры задач, эффективно решаемых с помощью АОП.

Самостоятельная работа студентов (26 часов)

Перечень занятий на СРС	Объем, час
Подготовка к лабораторным работам.	16
Изучение теоретического материала, не освещаемого на лекциях	10

5. Перечень учебной литературы

1. Абельсон, Х. Структура и интерпретация компьютерных программ: [пер. с англ.] / Москва: Добросвет : КДУ, 2011. – 608с., ISBN 978-5-98227-829-6 (10 экз.)

2. Фултон, Х. Программирование на языке Ruby: руководство / Москва : ДМК Пресс, 2007. – 684с., ISBN 5-94074-357-9 (1 экз.)

6. Перечень учебно-методических материалов по самостоятельной работе обучающихся

7. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины.

Для освоения дисциплины используются следующие ресурсы:
- электронная информационно-образовательная среда НГУ (ЭИОС);
- образовательные интернет-порталы;
- информационно-телекоммуникационная сеть Интернет.

7.1 Современные профессиональные базы данных

Не используются.

7.2. Информационные справочные системы

1. Электронные ресурсы Web of Science Core Collection (Thomson Reuters Scientific LLC.), Journal Citation Reports + ESI

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине.

Для обеспечения реализации дисциплины используется стандартный комплект программного обеспечения (ПО), включающий регулярно обновляемое лицензионное ПО Windows и MS Office, среда разработки Microsoft Visual Studio.

Использование специализированного программного обеспечения для изучения дисциплины не требуется.

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

Для реализации дисциплины используются специальные помещения:

1. Учебные аудитории для проведения занятий лекционного типа, лабораторных занятий, текущего контроля и промежуточной аттестации.

2. Помещения для самостоятельной работы обучающихся.

Учебные аудитории укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

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

Материально-техническое обеспечение образовательного процесса по дисциплине для обучающихся из числа лиц с ограниченными возможностями здоровья осуществляется согласно «Порядку организации и осуществления образовательной деятельности по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья в Новосибирском государственном университете».

10. Оценочные средства для проведения текущего контроля и промежуточной аттестации по дисциплине

10.1 Порядок проведения текущего контроля и промежуточной аттестации по дисциплине

Текущий контроль

Текущий контроль осуществляется в ходе семестра путем опроса в начале каждой лекции по материалам предыдущей лекции, а также проведения опроса студентов в начале каждого лабораторного занятия по теме, рассмотренной на предыдущем занятии.

Промежуточная аттестация

Освоение компетенций оценивается согласно шкале оценки уровня сформированности компетенции. Положительная оценка по дисциплине выставляется в том случае, если заявленная компетенция ПК-2 сформирована не ниже порогового уровня в части, относящейся к формированию способности использовать специализированные знания в области технологии разработки программного обеспечения в профессиональной деятельности.

Окончательная оценка работы студента в течение семестра происходит на дифференцированном зачете. Дифференцированный зачет в конце семестра в устной форме. Вопросы подбираются таким образом, чтобы проверить уровень сформированности компетенции ПК-2.

Вывод об уровне сформированности компетенций принимается преподавателем. Каждый вопрос билета оценивается от 0 до 5 баллов. Положительная оценка ставится, когда все компетенции освоены не ниже порогового уровня. Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Соответствие индикаторов и результатов освоения дисциплины

Таблица 10.1

Индикатор	Результат обучения по дисциплине	Оценочные средства
ПК -2.1. Проводит научные изыскания в избранной области экспериментальных и/или теоретических физических исследований с помощью современной аппаратуры и информационно-телекоммуникационных технологий в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Знать** методы проектирования программных средств вычислительной техники; методы и алгоритмы объектно-ориентированного, функционального, аспектно-ориентированного программирования**.**	Лабораторные занятия, дифференцированный зачет.

ПК -2.2. Применяет теоретические основы и базовые представления научного исследования в выбранной области фундаментальной и/или экспериментальной физики в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Уметь** комбинировать различные языки и системы программирования, а также методы проектирования с целью оптимального решения поставленных задач; Использовать типовые программные продукты, ориентированные на решение научных, проектных и технологических задач. **Владеть** рядом современных функциональных, динамических и аспектно-ориентированных языков, а также соответствующими им методами проектирования; навыками самостоятельной научно-исследовательской деятельности.	Лабораторные занятия, дифференцированный зачет.

10.2 Описание критериев и шкал оценивания индикаторов достижения результатов обучения по дисциплине «Технология разработки программного обеспечения».

Таблица 10.2

Критерии оценивания результатов обучения	Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Уровень освоения компетенции			
		Не сформирован (0 баллов)	Пороговый уровень (3 балла)	Базовый уровень (4 балла)	Продвинутый уровень (5 баллов)
1	2	3	4	5	6
Полнота знаний	ПК 2.1	Уровень знаний ниже минимальных требований. Имеют место грубые ошибки.	Демонстрирует общие знания базовых понятий по темам/разделам дисциплины. Допускается значительное количество негрубых ошибок.	Уровень знаний соответствует программе подготовки по темам/разделам дисциплины. Допускается несколько негрубых/ несущественных ошибок. Не отвечает на дополнительные вопросы.	Уровень знаний соответствует программе подготовки по темам/разделам дисциплины. Свободно и аргументированно отвечает на дополнительные вопросы.
Наличие умений	ПК 2.2	Отсутствие минимальных умений. Не умеет решать стандартные задачи. Имеют место грубые ошибки.	Продемонстрированы частично основные умения. Решены типовые задачи. Допущены негрубые ошибки.	Продемонстрированы все основные умения. Решены все основные задания с негрубыми ошибками или с недочетами.	Продемонстрированы все основные умения. Решены все основные задания в полном объеме без недочетов и ошибок.

Наличие навыков (владение опытом)	ПК 2.2	Отсутствие владения материалом по темам/разделам дисциплины. Нет навыков в решении стандартных задач. Наличие грубых ошибок.	Имеется минимальный набор навыков при решении стандартных задач с некоторыми недочетами.	Имеется базовый набор навыков при решении стандартных задач с некоторыми недочетами.	Имеется базовый набор навыков при решении стандартных задач без ошибок и недочетов. Продемонстрированы знания по решению нестандартных задач.

10.3 Типовые контрольные задания и материалы, необходимые для оценки результатов обучения

Пример задания для решения

1. Задан набор символов и число **n**. Опишите функцию, которая возвращает список всех строк длины **n**, состоящих из этих символов и не содержащих двух одинаковых символов, идущих подряд. Например, для символов 'a', 'b', 'c' и n=2 результат должен быть ("ab" "ac" "ba" "bc" "ca" "cb") с точностью до перестановки. Не допускается использовать циклы или рекурсию.

Примеры вопросов для изучения

- Интерпретация и компиляция. AOT-, JIT- компиляция. Компиляция в динамических языках.
- Виды типизации. Преимущества и недостатки различных видов типизации. Типизация в динамических языках.
- Семантика регулярных выражений. Основные операции с регулярными выражениями.

Вопросы для дифференцированного зачета

1. Основные характеристики и области применения динамических языков.
2. Интерпретация и компиляция. AOT-, JIT- компиляция. Компиляция в динамических языках.
3. Виды типизации. Преимущества и недостатки различных видов типизации. Типизация в динамических языках.
4. Язык Ruby, классификация, основные реализации. Встроенные структуры данных.
5. Семантика регулярных выражений. Основные операции с регулярными выражениями.
6. Объектная модель Ruby.
7. Динамическое изменение объектной модели в Ruby: элементы Meta-Object Protocol.
8. Основные положения функциональной парадигмы программирования. Неподвижное состояние, преимущества и недостатки.
9. Понятие лексического контекста. Замыкания.
10. Побочные эффекты. Чистые функции. Преимущества и недостатки.
11. Функции как объекты первого класса. Функции высших порядков: функционал, оператор. Примеры. Операторы композиции и частичного применения (каррирования).
12. Основные функции преобразования коллекций: map, reduce, filter. Отличия от прямой итерации по коллекции. Примеры использования (на Ruby или Clojure).
13. Язык Clojure. Концепция LISP: код как данные, связь с АСД. Модель компиляции/исполнения. REPL.
14. Встроенные типы и структуры данных Clojure. Основные операции.

15. Основные управляющие структуры Clojure: вызов функции, ветвления, цикл. Связь рекурсии и цикла. Хвостовая рекурсия в Clojure. Императивные управляющие структуры: последовательное исполнение, doseq.
16. Генератор for в Clojure, связь с map/reduce/filter.
17. Мемоизация. Область и примеры применения.
18. Реорганизующее присваивание (destructuring) в Clojure. Использование при объявлении/вызове функции, в let.
19. Отложенные вычисления на примере Clojure. Ленивые последовательности, delay. Основные операции над ленивыми последовательностями.
20. Потоки данных. Моделирование состояния с помощью потоков. Примеры использования. Бесконечные потоки.
21. Квотирование (quote). Виды квотирования в Clojure. Обратные операции: unquote, eval.
22. Специальные формы. Макросы. Модель исполнения. Применение макросов.
23. Особенности разрешения символов в eval и макросах. Внутренние переменные в макросах.
24. Разделение ответственностей. Принцип KISS. Связь с модульностью и абстракцией.
25. Ответственности 2-го класса (cross-cutting concerns). Примеры. Способы разделения ответственностей.
26. Инверсия управления (принцип Голливуда). Примеры применения.
27. Внедрение зависимостей (Dependency Injection, DI). Элементарное DI. DI с использованием контейнера. Связь с порождающими шаблонами проектирования.
28. Формы управления параллелизмом без блокировок.
29. Atomic-типы. Atomic-ссылка в Clojure, основные операции. Агенты. Future, promise.
30. Транзакционная память. Multi-Version Concurrency Control. Реализация в Clojure: алгоритм выполнения транзакции, изоляция транзакций.
31. Понятие распределенной транзакции. CAP-теорема. Транзакции типа Copy-Modify-Merge.
32. Формы полиморфизма. Полиморфизм в динамических языках. Принцип подстановки Барбары Лисков (строгая формулировка). Интерпретация в контрактном программировании.
33. Полиморфизм в иерархиях с одиночным и множественным наследованием. Комбинация методов на примере CLOS.
34. Обобщенные функции и посылка сообщений. Диспетчеризация по нескольким параметрам. Примеры использования.
35. Вспомогательные методы. Комбинация в иерархии наследования на примере CLOS. Примеры использования для разделения ответственностей.
36. Command-Query Separation. Применение в функциональных и императивных объектных моделях.
37. Аспектно-ориентированное программирование. Перехват, способы реализации. Применение для разделения ответственностей.
38. Понятие динамического лексического контекста. Реализация в Clojure. Связь с АОП. Применение для разделения ответственностей.

Пример билета к дифференцированному зачету

1. Интерпретация и компиляция. AOT-, JIT- компиляция. Компиляция в динамических
2. Аспектно-ориентированное программирование. Перехват, способы реализации. Применение для разделения ответственностей.

Форма экзаменационного билета представлена на рисунке

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Новосибирский национальный исследовательский государственный университет» (Новосибирский государственный университет, НГУ)

Физический факультет

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № _____

1. ……
2. ……

Составитель _____ /Ф.И.О. преподавателя/
(подпись)

«____»_____20 г.

Оценочные материалы по промежуточной аттестации, предназначенные для проверки соответствия уровня подготовки по дисциплине требованиям СУОС, хранятся на кафедре-разработчике РПД в печатном и электронном виде.

Лист актуализации рабочей программы
по дисциплине «Технология разработки программного обеспечения»
по направлению подготовки 03.04.02 Физика
Профиль «Информационные процессы и системы»

№	Характеристика внесенных изменений (с указанием пунктов документа)	Дата и № протокола Учёного совета ФФ НГУ	Подпись ответственного