МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ,
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ГОУ ВПО Кыргызско-Российский Славянский университет
имени первого Президента Российской Федерации Б.Н. Ельцина

УТВЕРЖДАЮ
декан факультета

2021 г.

Профессиональный английский язык

рабочая программа дисциплины (модуля)

Закреплена за кафедрой **Иностранных языков**

Учебный план g09040440_22_0 пи_рпис.plx
Направление подготовки 09.04.04 - РФ, 710400 - КР Программная инженерия
Магистерская программа "Разработка программно-информационных систем"

Квалификация **магистр**

Форма обучения **очная**

Общая трудоемкость **3 ЗЕТ**

Часов по учебному плану 108
в том числе:
аудиторные занятия 32
самостоятельная работа 75,8

Виды контроля в семестрах:
зачеты с оценкой 2

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	2 (1.2)		Итого	
Недель	17			
Вид занятий	УП	РП	УП	РП
Практические	32	32	32	32
Контактная работа в период теоретического обучения	0,2	0,2	0,2	0,2
В том числе инт.	4	4	4	4
Итого ауд.	32	32	32	32
Контактная работа	32,2	32,2	32,2	32,2
Сам. работа	75,8	75,8	75,8	75,8
Итого	108	108	108	108

Программу составил(и):
старший преподаватель , Кульматова Ч.Э.; старший преподаватель, Любимова Н.А. _____

Рецензент(ы):
к.пед.н., доцент, Юрченко М.Г. _____

Рабочая программа дисциплины
Профессиональный английский язык

разработана в соответствии с ФГОС 3++:
Федеральный государственный образовательный стандарт высшего образования - магистратура по направлению подготовки 09.04.04 Программная инженерия (приказ Минобрнауки России от 19.09.2017 г. № 932)

составлена на основании учебного плана:
Направление подготовки 09.04.04 - РФ, 710400 - КР Программная инженерия
Магистерская программа "Разработка программно-информационных систем"
утвержденного учёным советом вуза от 28.06.2022 протокол № 11.

Рабочая программа одобрена на заседании кафедры
Иностранных языков

Протокол от 6 . 09 2021 г. № 1
Срок действия программы: 2021-2023 уч.г.
Зав. кафедрой к.пед.н., Юрченко М.Г.

Визирование РПД для исполнения в очередном учебном году

Председатель УМС

__ _____ 2022 г.

Рабочая программа пересмотрена, обсуждена и одобрена для исполнения в 2022-2023 учебном году на заседании кафедры
Иностранных языков

Протокол от __ _____ 2022 г. № __
Зав. кафедрой к.пед.н., Юрченко М.Г.

Визирование РПД для исполнения в очередном учебном году

Председатель УМС

__ _____ 2023 г.

Рабочая программа пересмотрена, обсуждена и одобрена для исполнения в 2023-2024 учебном году на заседании кафедры
Иностранных языков

Протокол от __ _____ 2023 г. № __
Зав. кафедрой к.пед.н., Юрченко М.Г.

Визирование РПД для исполнения в очередном учебном году

Председатель УМС

__ _____ 2024 г.

Рабочая программа пересмотрена, обсуждена и одобрена для исполнения в 2024-2025 учебном году на заседании кафедры
Иностранных языков

Протокол от __ _____ 2024 г. № __
Зав. кафедрой к.пед.н., Юрченко М.Г.

Визирование РПД для исполнения в очередном учебном году

Председатель УМС

__ _____ 2025 г.

Рабочая программа пересмотрена, обсуждена и одобрена для исполнения в 2025-2026 учебном году на заседании кафедры
Иностранных языков

Протокол от __ _____ 2025 г. № __
Зав. кафедрой к.пед.н., Юрченко М.Г.

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ	
1.1	1.1 Целью освоения дисциплины «Профессиональный иностранный язык» является:
1.2	1.2 приобретение магистрантами коммуникативной компетенции, уровень которой позволяет практически использовать иностранный язык как в профессиональной (в производственной и научной) деятельности, так и в целях дальнейшего самообразования

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП	
Цикл (раздел) ООП:	Б1.О
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Для успешного изучения дисциплины «Профессиональный иностранный язык» магистрант должен обладать обязательными «входными» знаниями по дисциплине «Иностранный язык», полученными в ходе освоения основной образовательной программы по направлению «Программная инженерия» (бакалавриат или специалист).
2.2	**Дисциплины и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Научно-исследовательская работа магистранта

3. КОМПЕТЕНЦИИ ОБУЧАЮЩЕГОСЯ, ФОРМИРУЕМЫЕ В РЕЗУЛЬТАТЕ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)	
УК-4: Способен применять современные коммуникативные технологии, в том числе на иностранном(ых) языке(ах), для академического и профессионального взаимодействия	
Знать:	
Уровень 1	правила и закономерности личной и деловой устной и письменной коммуникации; современные коммуникативные технологии на русском и иностранном языках; существующие профессиональные сообщества для профессионального взаимодействия.
Уметь:	
Уровень 1	применять на практике коммуникативные технологии, методы и способы делового общения для академического и профессионального взаимодействия.
Владеть:	
Уровень 1	методикой межличностного делового общения на русском и иностранном языках, с применением профессиональных языковых форм, средств и современных коммуникативных технологий.

В результате освоения дисциплины обучающийся должен

3.1	**Знать:**
3.1.1	терминологию по своей специальности на иностранном языке; специфику и приёмы перевода различных грамматических конструкций; особенности ведения профессиональной и научной деятельности; коммуникативные правила поведения в ситуациях межкультурного профессионального общения; требования, предъявляемые к оформлению научных трудов, принятые в международной практике; стилистические особенности представления результатов профессиональной и научной деятельности в устной и письменной форме на иностранном языке; основные современные компьютерные технологии, используемые для сбора, анализа и обработки данных.
3.2	**Уметь:**
3.2.1	осуществлять устную коммуникацию в монологической и диалогической форме научной направленности (доклад,сообщение, презентация, дебаты, круглый стол); писать научные аннотации, эссе, тезисы, рефераты; читать оригинальную литературу на иностранном языке в соответствующей отрасли знаний; правильно выбирать адекватные языковые средства перевода научной литературы; анализировать, обобщать и интерпретировать информацию по своей специальности на иностранном языке; оформлять извлеченную из иностранных источников информацию в виде перевода, реферата, аннотации; извлекать информацию из текстов, прослушиваемых в ситуациях межкультурного научного общения и профессионального (доклад, лекция, интервью, дебаты, и др.); использовать этикетные формы научно - профессионального общения; четко и ясно излагать свою точку зрения по научной проблеме на иностранном языке; производить различные логические операции (анализ, синтез, установление причинно-следственных связей, аргументирование, обобщение и вывод,комментирование); понимать и оценивать чужую точку зрения, стремиться к сотрудничеству, достижению согласия, выработке общей позиции в условиях различия взглядов и убеждений; следовать основным нормам, принятым в научном общении на государственном и иностранном языках; применять современные коммуникативные технологии для решения поставленных задач в своей профессиональной деятельности.
3.3	**Владеть:**

3.3.1	терминологическим аппаратом на иностранном языке по своей специальности; навыками и умениями устной и письменной речи на иностранном языке, позволяющими достаточно свободно общаться с носителями языка; иметь опыт обработки большого объема иноязычной информации с целью подготовки реферата; иметь опыт оформления заявок на участие в международной конференции; иметь опыт написания работ на иностранном языке для публикации в зарубежных журналах; следовать основным нормам, принятым в научном общении на государственном и иностранном языках; применять новые информационные и коммуникативные технологии для решения поставленных задач в своей профессиональной деятельности.

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Компетен-ции	Литература	Инте ракт.	Пр. подг.	Примечание
	Раздел 1. Knowledge Management							
1.1	Тема: Менеджмент знаний. 1) Вдовичев А. В., Оловникова Н. Г. Текст «12 Principles of Knowledge Management», стр. 76-77. Ознакомительное чтение текста. Лексико-грамматический анализ текста. 2) Беседа на тему «If I were a Minister of Education». /Пр/	2	2		Л1.1Л2.1 Л2.2 Э1 Э5 Э9			
1.2	Повторение условных предложений и предложений с придаточными времени. /Ср/	2	2		Л1.1			
1.3	Вдовичев А. В., Оловникова Н. Г. Текст «В. В. Путин: О развитии образования в Российской Федерации», стр. 64-66. Перевод текста на английский язык. /Ср/	2	2		Л1.1			
1.4	Тема: «Наука в нашей повседневной жизни» 1. Мозговой штурм: «Какие научные открытия Вы считаете самыми полезными для нашей повседневной жизни?» 2. Обмен мнениями о высказываниях различных ученых о вкладе науки в совершенствование жизни человека (Вдовичев А., с. 106-107) 3. Лексико-грамматический анализ текста: «How Research in Chemistry Has Improved Daily Life». Выполнение послетекстовых заданий. Грамматика. Сложное подлежащее. Сложное дополнение. Предлоги. Артикли /Пр/	2	2		Л1.1 Э2 Э4 Э7 Э9	2		Мозговой штурм
1.5	Написание эссе о роли и значении образования и знаний в жизни человека. /Ср/	2	2		Л1.1			

1.6	Тема: Электронная почта. 1) Esteras S. R., Fabre E. M. Professional English in Use: ICT. Unit 22 «Email», стр. 54. Подготовка текста к аннотированию. 2) Выполнение упражнений, стр. 55. /Пр/	2	2		Л2.2 Э1 Э2 Э5 Э6 Э7 Э9			
1.7	Esteras S. R., Fabre E. M. Professional English in Use: ICT. Unit 23 «The WWW», стр. 56-57. /Ср/	2	2		Л2.2			
1.8	Подготовка текста «The WWW» для пересказа. Работа с новой лексикой, выполнение упражнений, аннотирование текста. /Ср/	2	2		Л2.2			
1.9	Подготовка материалов о научных достижениях в области разработки программно-информационных систем для внеаудиторного чтения. Подготовка мини-реферата. /Ср/	2	4		Э1 Э5 Э6 Э7 Э8 Э9			
1.10	ЗЕТ-1 Защита мини-реферата по предложенным темам. /Пр/	2	2		Л1.1			
1.11	Вдовичев А. В., Оловникова Н. Г. Текст «Destructive Creativity in Scientific Research», стр. 88-90. Перевод и пересказ текста. Написание краткого изложения (summary) текста. /Ср/	2	4		Л1.1			
1.12	Вишневская Г.А., Коркмазова Н. И., Ч. Э. Кульматова, А. Н. Тухтарова, А. К. Успеева Методическая разработка по внеаудиторному чтению для студентов 2-3 курсов специальности ИВТ. Текст «A Case Study: MSDOS», стр. 22-24. Изучение новой лексики, подготовка текста для пересказа. Выполнение упражнений, стр. 24-26. /Ср/	2	2		Л1.1Л2.1			
1.13	Тема: «Основные принципы магистерской работы» 1. «A suggested thesis structure». 2. Изучение вопросов, касающиеся структуры диссертационного исследования за рубежом, сопоставление их с требованиями в КР и РФ, используя двухчастный дневник. 3. Описание структуры научной работы магистрантов /Пр/	2	2		Л1.1			
1.14	Поиск дополнительной информации по теме «Технические достижения как гордость нации». /Ср/	2	2		Л1.1Л2.2			
	Раздел 2. Тема: Perspectives on Science and Technology							

2.1	Тема: Перспективы развития науки. 1) Обмен мнениями по вопросам образования: 1.«If you were the Minister of Education, which priorities in knowledge building would you take?». Приведите свои аргументы. 2. «Do you think there is a well-built knowledge management and knowledge quality assurance system in the educational area of Russia and Kyrgyzstan? What should be improved and how?». 2) Дискуссия на тему «Quo Vadis Homo Futuris?». 3) Проверка внеаудиторного чтения. /Пр/	2	2		Л1.1			
2.2	Повторение инфинитивных конструкций (инфинитив цели). /Ср/	2	4		Л1.1			
2.3	Подготовка внеаудиторного чтения (подбор текстов по специальности, чтение, перевод, изучение лексики, составление вопросов, пересказ). /Ср/	2	4		Л1.1			
2.4	Тема: Научные открытия или непредвиденные последствия? 1) Презентация кратких устных сообщений о достижениях в области разработки программно-информационных систем. 2) Вдовичев А. В., Оловникова Н. Г. Текст «Nanotechnology», стр. 101-102. Ознакомительное чтение текста. Выполнение лексико-грамматических упражнений, стр. 102-105. 3) Подготовка 5-минутной речи по теме урока, выступление с последующим переводом речи на английский язык. /Пр/	2	2		Л1.1Л2.2			
2.5	Изучение материалов для подготовки доклада по специальности. /Ср/	2	4		Э4 Э5 Э6 Э7 Э8 Э9			
2.6	Подготовка доклада по специальности. /Ср/	2	4		Л1.1			
2.7	Написание эссе на одну из тем: «The Discovery I Am Proud of», «Science Will Never End». /Ср/	2	2		Л1.1			
2.8	ЗЕТ-2. Защита докладов по заданным темам. /Пр/	2	2					
2.9	Esteras S. R., Fabre E. M. Professional English in Use: ICT. Professional English in Use: ICT. Unit 25 «Chatting and Video Conference», стр. 56-57. /Ср/	2	4		Л2.2 Э3 Э4 Э6 Э8			

2.10	Подготовка внеаудиторного чтения (подбор текстов по теме, чтение, перевод, изучение лексики, составление вопросов, пересказ). Аннотирование текста. /Ср/	2	4		Л2.2 Э1 Э2 Э3			
	Раздел 3. Тема: Science in Everyday Life							
3.1	Тема: Наука в повседневной жизни 1) Вдовичев А. В., Оловникова Н. Г. Текст «Science in Our Everyday Life», стр. 106. Ознакомительное чтение текста. Выполнение лексических упражнений, стр. 106-107. 2) Написание 5-10 предложений о внедрении и экспериментальном использовании научных изобретений. 3) Проверка внеаудиторного чтения. /Пр/	2	2		Л1.1			
3.2	Написание эссе на тему «Science in Everyday Life». /Ср/	2	2		Л1.1 Э1 Э2 Э4 Э6			
3.3	Подготовка внеаудиторного чтения (подбор текстов по теме, чтение, перевод, изучение лексики, составление вопросов, пересказ). Аннотирование текста. /Ср/	2	2		Л1.1			
3.4	Тема: Веб дизайн. 1) Esteras S. R., Fabre E. M. Professional English in Use: ICT. Unit 24 «Web Design», стр. 58. Ознакомительное чтение текста, перевод и обсуждение текста. 2) Выполнение упражнений, стр. 59. 3) Проверка внеаудиторного чтения. /Пр/	2	2		Л1.1Л2.2			
3.5	Повторение пассивного залога, условных предложений и предложений с придаточными времени, инфинитивных конструкций (инфинитив цели). /Ср/	2	2		Л1.1			
3.6	Тема: Нанотехнологии 1)Вдовичев А. В., Оловникова Н. Г. стр. 101-105. Чтение текста, выполнение лексическо -грамматических упражнений. 2) Подготовка 5-8 минутной речи по теме урока, выступление с последующим переводом речи на английский язык. /Пр/	2	2		Л1.1			

3.7	Подготовка кратких сообщений, небольших текстов на тему «Технологизация современного человека» для обсуждения. Чтение, перевод, изучение новой лексики. /Ср/	2	2		Л1.1			
3.8	Вдовичев А. В., Оловникова Н. Г. Текст «Applications of Nanotechnology», стр. 125-126. Чтение, перевод, изучение новой лексики, пересказ и аннотирование текста. /Ср/	2	2		Л1.1			
3.9	ЗЕТ-3. Написание эссе на тему «Software and Data Systems in Everyday Life». /Пр/	2	2		Л1.1			
3.10	Вдовичев А. В., Оловникова Н. Г. Текст «The Magical Number Seven, Plus or Minus Two», стр. 129-130. Перевод, пересказ и аннотирование текста. Выполнение грамматического задания, стр. 129. /Ср/	2	2		Л1.1			
3.11	Подготовка внеаудиторного чтения (подбор текстов по специальности, чтение, перевод, изучение лексики, составление вопросов, пересказ). Аннотирование текста. /Ср/	2	2		Л1.1			
3.12	Тема: Internet Security Santiago Remacha Esteras, Elena Marco Fabre “Professional English in Use” ICT, unit 26. Лексико-грамматический анализ текста. Аннотирование текста. /Пр/	2	2		Л2.2 Э1 Э3 Э5 Э7 Э8			
3.13	Вдовичев А. В., Оловникова Н. Г. Текст «Инновационное образование: вызовы и решения», стр. 114-117. Письменный перевод текста со словарём. /Ср/	2	4		Л1.1			
3.14	Тема: Health Safety in the Technological Era. 1) Вдовичев А. В., Оловникова Н. Г. Текст «Scientists Warn US Congress of Cancer Risk for Cell Phone Use», стр. 126-128. Ознакомительное чтение текста, работа с новой лексикой. 2) Групповая работа: представление аргументов за и против на тему «Технологизация современного человека». 3) Проверка внеаудиторного чтения. /Пр/	2	2		Л1.1Л2.2			
3.15	Аннотирование текста «Инновационное образование: вызовы и решения», стр. 114-117. /Ср/	2	2		Л1.1Л2.2			

3.16	Вдовичев А. В., Оловникова Н. Г. Текст «A Suggested Thesis Structure», стр. 120-125. Самостоятельное изучение материала о структуре дипломных работ и диссертаций. /Ср/	2	2		Л1.1			
3.17	Тема: «Разработка программного обеспечения» 1. Круглый стол по проблемам связанным с разработкой программного обеспечения. 2. Презентация докладов по проблеме 3. Обсуждение докладов и аргументации выдвинутых магистрантами положений. 4. Выработка общей концепции по проблеме (Карта концепций)	2	2		Л1.1Л2.2 Э1 Э3 Э5 Э7 Э8 Э9	2		Круглый стол
3.18	Подготовка внеаудиторного чтения по теме: «The Vision of the Software Development» /Ср/	2	2		Э1 Э4 Э6 Э7 Э9			
3.19	Письменный перевод текста по специальности с английского на русский язык со словарём. /Пр/	2	2		Л1.1			
3.20	Поиск дополнительной информации и подготовка сообщений о выдающихся людях и достижениях в области разработки программно-информационных систем. /Ср/	2	2		Л1.1			
3.21	Написание краткого изложения (summary) текста по специальности для внеаудиторного чтения. /Ср/	2	1,8		Л1.1			
3.22	/КрТО/	2	0,2					
3.23	/ЗачётСОц/	2						

5. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

5.1. Контрольные вопросы и задания

ФОС:
Знать:
1. Опишите структуру выделенных предложений
2. Вставьте пропущенные артикли.
3. Объясните употребление артиклей в данном тексте.
4. Составьте предложение с указанными словами.
5. Соедините указанные слова с их дефинициями.
6. Дайте толкование слов и словосочетаний из области научного знания и специальности.
7. Составьте предложения с указанными словами и словосочетаниями из области научного знания и специальности.
8. Объясните употребление времен в указанных предложениях
9. Составьте вопросительные и отрицательные конструкции выделенных предложений.
10. Расставьте правильно времена в указанном тексте
11. Проведите коррекцию текста
12. Найдите в тексте предложения в страдательном залоге.
13. Переделайте предложения в страдательные конструкции
14. Проведите лексико- грамматический анализ текста
15. Определите стиль предложенного текста
16. Измените предложения с реальным условием в нереальные, объясните необходимую замену слов.
17. Укажите функции причастий, деепричастий в составе указанного предложения.
18. Найдите предложения со сложным дополнением в тексте.
19. Употребите сложное дополнение в предложенных предложениях.

20. Найдите герундии в тексте.
21. Вставьте пропущенные инфинитивы или герундии в текст.
22. Переведите прямую речь в косвенную в указанном тесте или диалоге с учетом согласования времен.
23. Вставьте пропущенные предлоги. Объясните их употребление.
24. Укажите значение и особенности употребления лексики в рамках изученных тем.
25. Объясните значение и особенности употребления специальной терминологии и профессиональной лексики в рамкахизученных тем.
26. Укажите особенности построения эссе, научной статьи, реферата, диссертационного исследования.
27. Перечислите характерные особенности научного и научно-популярного стиля изложения.
28. Назовите основные правила и законы подготовки и проведения успешной презентации
29. Перечислите основные правила ведения дискуссий, обмена мнениями для успешной коммуникации.
30. Открытые тесты по грамматике.
Уметь:
1. Составьте план беседы, доклада, научного реферата.
2. Составьте тезисы предложенной научной статьи: “ Наука в нашей жизни”.
3. Напишите аннотацию на предложенную статью.
4. Подготовьте устные логически точные аргументированные краткие и подробные сообщения по вопросам науки испециальности: «Инновационные технологии в образовании», «Последние тенденции в программной инженерии»,«Интеллектуальное право и собственность »,« Искусственный интеллект»« История развития компьютерных технологий».
5. Обоснуйте актуальность научной тематики статьи, монографии по специальности
6. Изложите свой взгляд на основную проблему статьи, научного исследования по специальности
7. Обдумайте правильные выводы и подведите итоги осуществленной коммуникации в научно-профессиональной сфере
8. Переведите устно и письменно тексты с русского языка на английский и с английского языка на русский поспециальности.
9. Подберите материал для участия в дискуссии на указанную тему, учитывая правила ведения дискуссий (аргументация,внимание к чужому мнению, вежливого выражения согласия и несогласия, умения подводить итоги дискуссии
10. Подготовьтесь к презентации выступления по вопросам роли математики в компьютерных вычислениях.
11. Передайте основное содержание прочитанного текста
12. Выделите основную идею прочитанного текста
13. Выделите основную идею просмотренного фильма (фрагмента фильма) “Family Album USA”
14. Сформулируйте свое мнение по прочитанному тексту. «The Internet»
15. Подготовьтесь к обмену мнениями по вопросам профессии важности теории игр и её практического применения дляразрешения сложных проблем в программировании.
16. Выразите свое мнение по просмотренному фильму.
17. Выразите свое мнение на заданную тему по различным типам программных систем и их эффективности.
18. Составьте аннотацию к прочитанному тексту.
19. Сформулируйте вопросы к прочитанному тексту, докладу или сообщению
20. Составьте фактологические вопросы к предложенному тексту
21. Задайте концептуальные вопросы к предложенному тексту.
22. Заполните словарную карту с указанными словами.
23. Заполните таблицу новых слов по прочитанному тексту.
24. Проведите лексико-грамматический анализ выделенных предложений в тексте.
25. Выберите из предложенных значений незнакомых слов одно, соответствующее контексту.
26. Выберите из предложенных значений незнакомых слов одно, соответствующее контексту.
27. Соберите и систематизируйте материал для участия в круглом столе на указанную тему: «The Evolution of SoftwareArchitecture».
28. Подготовьте обзор научного журнала по специальности (авторы, проблематика, актуальность материала, доступностьизложения материала, степень новизны)
29. Подготовьте текст газетной статьи по итогам проведенного круглого стола, дискуссии.
30. Систематизируйте найденный материал для выпускного реферата по научной работе магистранта. Составьте планработы, определите цели и задачи реферата, обоснуйте актуальность, степень новизны, практическую и научнуюзначимость своего научного исследования.
31. Используя двухчастный дневник и диаграмму Венна, сопоставьте структуры и требования к диссертационномуисследованию за рубежом с требованиями в КР и РФ.
32. Опишите структуру своей научной работы.
Владеть:
1. Обменяйтесь мнениями по вопросам науки и научного знания, образования и его совершенствования, места науки вжизни человека, о современном состоянии науке и последних изменениях в структуре и деятельности Академии наук КР и РФ
2. Примите участие в обсуждении определений данных учеными понятию экономика и экономическая наука, эволюцииданных понятий, а также основных проблем, поднятых в читаемых статьях.
3. Подготовьте сообщения на темы:«Выдающиеся программисты мира», «Выдающиеся программисты Киргизстана и России», «Математическоепрограммирование», «Кто такой разработчик?», «Роль позитивного мышления в научном познании», «Роль математики в нашей жизни», «Теория игр и их виды», «Инновационные методы в программной инженерии», «Наиболее актуальные проблемы современной науки»
4.Опишите структуру своей научной работы.

5.Проанализируйте сходства и различия в вопросах структуры диссертационного исследования зарубежом,сопоставьте их с требованиями в КР и РФ, используя двухчастный дневник. Описание структуры научной работы магистрантов
6.Составьте диалоги по изученным темам:«Основные субъекты программирования», «Выдающиеся программисты Киргизстана и России»,«Английский язык–язык международногообщения», «Процесс принятия решений в программировании»,«Роль позитивного мышления в научном познании», «Роль инновационных технологий в жизни общества»«Оптимизация», «Роль правительства в регулировании интернет технологий»,«Деструктивные идеи в научном сообществе»
7.Примите участие в дискуссии на указанную тему,учитывая правила ведения дискуссий(аргументация,внимание к чужому мнению, вежливого выражения согласия и несогласия, умения подводить итоги дискуссии):«Innovative Entrepreneurship in Modern Kyrgyzstan: A Dream or Reality?», «Современное состояние и новейшие научные достижения в профессиональной сфере магистрантов»
8. Примите участие в выработке общей концепции по проблеме будущего развития методов программирования.
9. Составьте карту концепций по актуальным вопросам специальности и области науки
10. Предоставьте к защите в аудитории презентации по вопросам математики,ее места и роли в жизни человека и специальности
11. Напишите отзыв на одну из прослушанных презентаций магистров
12. Напишите аннотацию на прочитанную статью (текст адаптирован).
13.Составьте тезисы устного выступления по указанной тематике
14. Напишите сочинение поизученным темам
15. Передайте краткое содержание прочитанной газетной статьи.
16. Передайте краткое содержание прочитанного текста по специальности.
17. Напишите мини-сочинение(эссе) на тему:эссе по вопросам эволюции науки,связанной с будущей профессией магистрантов, или смежных наук, уделяя особое внимание перспективам развития данной науки. Написание краткого эссе-размышления по вопросам по вопросам применение алгоритмов теории игр в информационных системах.
18. Выступите с обзором научного журнала по специальности (авторы, проблематика, актуальность материала, доступность изложения материала, степень новизны) 19. Составьте план прочитанного текста
20. Передайте содержание просмотренного фильма “Beautiful Mind”.
21. Дайте краткий письменный анализ сходства и различия героя художественного фильма “Beautiful Mind” с его прототипом - нобелевским лауреатом Джоном Нэшем.
22. Передайте содержание прослушанной аудиозаписи.
23. Напишите газетную заметку о проведенной дискуссии (от лица участника, слушателя)
24. Обоснуйтесвойвыборуказаннойдефиниции,цитаты.
25. Составьте устный рассказ по указанной теме, используя предложенные слова и словосочетания
26. Примите участие в мозговых штурмах по предложенным вопросам
27. Аргументируйте выбранные высказывания «за» и «против» предложенных определений эволюции информационных технологий.
28. Примите участие в круглом столе на тему: «Virtual Classrooms». Выступите с докладом, подготовьте аргументы для дискуссии и примите участие в обсуждении докладов.
29. Напишите и защитите выпускной реферат по научной работе магистрантов.

5.2. Темы курсовых работ (проектов)

Курсовые работы программой не предусмотрены

5.3. Фонд оценочных средств

1) Просмотровое чтение специализированного текста
2) Аннотирование специализированного текста
3) Письменный перевод специализированного текста со словарём
4) Темы эссе и презентаций 2-й семестр

5.4. Перечень видов оценочных средств

Письменный перевод специализированного текста со словарём, реферат, презентация, эссе, мини-доклад.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы, составители	Заглавие	Издательство, год
Л1.1	Вдовичев А.В, Оловникова Н.Г.	Английский для магистрантов и аспирантов: Учебно-методическое пособие	М: Флинта 2015

6.1.2. Дополнительная литература

	Авторы, составители	Заглавие	Издательство, год
Л2.1	Л. Васильева	Деловая переписка на английском языке	Москва .: Рольф 2001
Л2.2	Elena Marco Fabré, Santiago Remacha Esteras	Professional English in Use. ICT. : английский язык	Cambridge University Press 2007

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"		
Э1	Профессиональный английский язык	https://www.english4it.com/u nit/4/reading
Э2	Профессиональный английский язык	https://www.reddit.com/r/programming/
Э3	Профессиональный английский язык	https://news.ycombinator.com/news
Э4	Профессиональный английский язык	https://lenguainglessmvc.wik
Э5	Профессиональный английский язык	http://www.developerdotstar. com/mag/articles_top.html
Э6	Профессиональный английский язык	http://books.mcgraw-hill.com/computing/oleary/p
Э7	Профессиональный английский язык	http://www.indiana.edu/~tisj/readers/full-text/16-3%
Э8	Профессиональный английский язык	http://www.businessenglish s ite.com/business-english-
Э9	Профессиональный английский язык	http://www.multimedia-english.com/videos/lesson/

6.3. Перечень информационных и образовательных технологий	
6.3.1 Компетентностно-ориентированные образовательные технологии	
6.3.1.1	Для успешного овладения курсом рекомендуется использовать следующие виды программного обеспечения:Microsoft Word, Power Point.
6.3.1.2	В ходе освоения дисциплины используются следующие виды учебной работы: практические занятия, дискуссии,ролевые игры, подготовка презентаций, исследовательская работа, индивидуальная и самостоятельная работа.
6.3.2 Перечень информационных справочных систем и программного обеспечения	
6.3.2.1	6.3.2.1. https://www.english4it.com/unit/4/reading
6.3.2.2	6.3.2.22. https://www.reddit.com/r/programming
6.3.2.3	6.3.2.33. https://news.ycombinator.com/news
6.3.2.4	6.3.2.44. http://www.developerdotstar.com/mag/articles_top.html
6.3.2.5	6.3.2.55. http://books.mcgraw-hill.com/computing/oleary/pdf/ole65985_ch01_web.pdf
6.3.2.6	6.3.2.66. http://www.indiana.edu/~tisj/readers/full-text/16-3%20kling.pdf
6.3.2.7	6.3.2.77. http://www.businessenglishsite.com/business-english-computers-it.html
6.3.2.8	6.3.2.88. http://www.multimedia-english.com/videos/lesson/english-vocabulary-for-esl-it-computing-web-2.0-business-english-pod-5090
6.3.2.9	6.3.2.99. http://ff.tu-sofia.bg/~bogi/France/SoftEng/books/Software%20Engineering-PH-Successful-Software-Development-2nd-Edition.pdf 6.3.2.1010. https://lenguainglessmvc.wikispaces.com/file/view/english-for-information-technology.pdf/592178992/english- for-information-technology.pdf

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)	
7.1	Учебные аудитории рассчитаны на 20 посадочных мест, оснащены интерактивной доской. В ходе занятий так жеиспользуются: ноутбук, проэктор, раздаточный материал (статьи для перевода), ролевые игры.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

Технологическая карта приложение 7
1.Советы по планированиюиорганизациивремени,необходимогодляизученияДисциплины.
Рекомендуется следующим образом организовать время, необходимое для изучения дисциплины:Изучение конспекта лекции в тот же день, после лекции - 10-15 минут.Изучение конспекта лекции за день перед следующей лекцией -10-15 минут.Изучение теоретического материала по учебнику и конспекту - 1 час в неделю.Подготовка к практическому занятию - 2 час.Всего в неделю - 3 часа 30 минут.
2.ОписаниепоследовательностидействийстудентаДля понимания материала и качественного его усвоения рекомендуется такая последовательность действий:
1.)Послепрослушиванияlекцииокончанияучебныхзанятий,приподготовкекзанятиямследующегодня,нужносначала просмотреть и обдумать текст лекции, прослушанной сегодня (10-15 минут).
2.)Приподготовкеклекциисследующегодня,нужнопросмотретьтекстпредыдущейлекции,подуматьотом,какаяможет быть тема следующей лекции (10-15 минут).
3.)Втечениенеделивыбратьвремя(1-час)дляработысрекомендуемойлитературойвбиблиотеке.
4.)При подготовке к практическимзанятиям следующегодня,необходимосначалапрочитатьосновныепонятияиподходы по теме домашнего задания. При выполнении упражнения или задачи нужно сначала понять, что требуется взадаче, какой теоретический материал нужно использовать, наметить план решения задачи.
3. Рекомендациипоработеслитературой.Теоретическийматериалкурсастановитсяболеепонятным,когдадополнительно к прослушиванию лекции и изучению конспекта, изучаются и книги. Легче освоить курс, придерживаясьодного учебника и

конспекта. Рекомендуется, кроме «заучивания» материала, добиться состояния понимания изучаемойтемы дисциплины. С этой целью рекомендуется после изучения очередного параграфа выполнить несколько простыхупражнений на данную тему.
4. Советыподготовкекрубежномуипромежуточномуконтролям.Дополнительнокизучениюконспектовлекции необходимо пользоваться учебником. Кроме «заучивания» материала, очень важно добиться состояния пониманияизучаемых тем дисциплины. С этой цепью рекомендуется после изучения очередного параграфа выполнить несколькоупражнений на данную тему. Кроме того, очень полезно мысленно задать себе следующие вопросы (и попробоватьответил, на них): о чем этот параграф?, какие новые понятия введены, каков их смысл?, что даст это на практике?. Приподготовке к промежуточному контролю нужно изучить теорию: определения всех понятий и подходы к оцениванию досостояния понимания материала.
5. Указанияпоорганизацииработысконтрольно-измерительнымиматериалами,повыполнениюдомашнихзаданий. При выполнении домашних заданий необходимо сначала прочитать основные понятия и подходы по темезадания. При выполнении упражнения или задачи нужно сначала понять, что требуется в задаче, какой теоретическийматериал нужно использовать, наметить план решения задачи, а затем приступить к расчетам и сделать качественныйвывод.
6. Рекомендации по написанию реферата.
1.)ТемарефератавыбираетсявсоответствиисВашимиинтересамиинеобязательнодолжнасоответствоватьприведенному ниже примерному перечню. Важно, чтобы в реферате: во-первых, были освещены как естественнонаучные,так и социальные стороны проблемы; а во-вторых, представлены как общетеоретические положения, так и конкретныепримеры. Особенно приветствуется использование собственных примеров из окружающей Вас жизни.
2.)Рефератдолженосновыватьсянапроработкенесколькихдополнительныхкосновнойлитературеисточников.Как правило, это специальные монографии или статьи.
3.)Планрефератадолженбытьавторским.Внемпроявляетсяподходавтора,егомнение,анализпроблемы.
4.)Рефератоформляетсяввидетекстаналистахстандартногоформата(А-4).Начинаетсяститульноголиста,вкотором указывается название вуза, учебной дисциплины, тема реферата, фамилия и инициалы студента, номеракадемической группы или название кафедры, год и географическое место местонахождения вуза. Затем следуетоглавление с указанием страниц разделов. Сам текст реферата желательно подразделить на разделы: главы, подглавы иозаглавить их. Приветствуется использование в реферате количественных данных и иллюстраций (графики, таблицы,диаграммы, рисунки).
5.)Завершаютрефератразделы"Заключение"и"Списокиспользованнойлитературы".Взаключениипредставленыосновные выводы, ясно сформулированные в тезисной форме и, обычно, пронумерованные.
6.)Списоклитературыдолженбытьсоставленвполномсоответствиесдействующимстандартом(правилами),включая особую расстановку знаков препинании. Для этого достаточно использовать в качестве примера любую книгуизданную крупными научными издательствами: "Наука", "Прогресс", "Мир", "Издательство МГУ" и др. Или приведенныйвыше список литературы. В общем случае наиболее часто используемый в нашей стране порядок библиографическихссылок следующий:Автор И.О. Название книги. Место издания: Издательство, Год издания. Общее число страниц в книге.Автор И.О. Название статьи //Название журнала Год издания. Там_. No_. Страницы от__до_.Автор И.О. Название статьи/Название сборника. Место издания: Издательство, Год издания. Страницы от___до_.
7. Основные требования к написанию эссеЭссе – это самостоятельная письменная работа на тему, предложенную преподавателем соответствующей дисциплины илисамостоятельно избранная студентом по проблематике читаемого курса. Цель написания эссе состоит в развитии навыковсамостоятельного творческого подхода к пониманию и осмыслению проблем научного знания, а также навыковписьменного изложения собственных мыслей и отношения к различным проблемам.
Требования к содержанию эссе
ВведениеВведение должно включать обоснование интереса к выбранной теме, ее актуальность или практическую значимость.Важно учесть, что заявленная тема должна быть адекватна раскрываемому в эссе содержанию, иначе говоря, не должнобыть рассогласования в названии и содержании работы.Основная частьОсновная часть предполагает последовательное, логичное и доказательное раскрытие заявленной темы эссе.
8. Подготовка к экзаменуЛюбой экзамен состоит из нескольких частей, на которых проверяются все навыки знания английского: письменную иустную речь, чтение, словарный запас, знание грамматики, произношение.
Для развития данных навыков рекомендуется следовать следующим указаниям:Навык чтения. Читайте тексты разной тематики: от газет и журналов до художественной и научно-популярной литературы.Так вы изучите новые слова из контекста, увеличите скорость чтения, при помощи зрительной памяти запомнитеправописание новых слов, освежите знания слов и фраз по конкретным темам. Типовое задание по чтению на экзамене —это вопросы на осмысление текста.Навык говорения. Типовое задание представляет собой спонтанную речь на заданную тему. Эту часть экзамена лучшевсего репетировать со своим преподавателем: он поможет вам исправить типовые ошибки и научит говорить естественно.Есть простое и очень эффективное упражнение для тренировки спонтанной речи: выберите любой предмет и старайтесьговорить о нем 1 минуту. Так вы научитесь говорить о чем угодно, а это именно то, что нужно вам на экзамене. Можно исамостоятельно работать над навыком говорения. Для этого попробуйте следующий метод. Возьмите интересныйвидеоролик, посмотрите его несколько раз, выучите субтитры. После этого включите видео без звука и попытайтесьпопасть в темп говорящего. Это упражнение поможет вам выучить полезную лексику и научиться бегло говорить по-английски. Кроме того, не забывайте читать вслух: вы улучшите свое произношение, поработаете над интонацией иправильным акцентом, научитесь формулировать свои мысли ясно и грамотно.

Контрольные задания для магистрантов по направлению «Программная инженерия» (ЕПИМ)

1 курс 2 семестр

ЗЕТ-1. Тексты по специальности для письменного перевода с английского на русский язык со словарём.

Текст 1: The Internet

Millions of people around the world use the Internet to search for and retrieve information on all sorts of topics in a wide variety of areas including the arts, business, government, humanities, news, politics and recreation. People communicate through electronic mail (e-mail), discussion groups, chat channels and other means of information exchange. They share information and make commercial and business transactions. All this activity is possible because tens of thousands of networks are connected to the Internet and exchange information in the same basic ways.

The World Wide Web (WWW) is a part of the Internet. But it's not a collection of networks. Rather, it is information that is connected or linked together like a web. You access this information through one interface or tool called a Web browser. The number of resources and services that are part of the World Wide Web is growing extremely fast. In 1996, there were more than 20 million users of the WWW, and more than half the information that is transferred across the Internet is accessed through the WWW. By using a computer terminal (hardware) connected to a network that is a part of the Internet, and by using a programme (software) to browse or retrieve information that is a part of the World Wide Web, the people connected to the Internet and World Wide Web through the local providers have access to a variety of information. Each browser provides a graphical interface. You move from place to place, from site to site on the Web by using a mouse to click on a portion of text, icon, or region of a map. These items are called hyperlinks or links. Each link you select represents a document, an image, a video clip or an audio file somewhere on the Internet. The user doesn't need to know where it is, the browser follows the link.

All sorts of things are available on the WWW. One can use the Internet for recreational purposes. Many TV and radio stations broadcast live on the WWW. Essentially, if something can be put into digital format and stored in a computer, then it's available on the WWW. You can even visit museums, gardens and cities throughout the world, learn foreign languages and meet new friends. And, of course, you can play computer games through the WWW, competing with partners from other countries and continents.

Just a little bit of exploring the World Wide Web will show you what a lot of use and fun it is.

Text 2: Software process

A software process is a set of related activities that leads to the production of a software product. These activities may involve the development of software from scratch in a standard programming language like Java or C. However, business applications are not necessarily developed in this way. New business software is now often developed by extending and modifying existing systems or by configuring and integrating off-the-shelf software or system components.

There are many different software processes but all must include four activities that are fundamental to software engineering:

1. Software specification The functionality of the software and constraints on its operation must be defined.

2. Software design and implementation The software to meet the specification must be produced.

3. Software validation The software must be validated to ensure that it does what the customer wants.

4. Software evolution The software must evolve to meet changing customer needs.

In some form, these activities are part of all software processes. In practice, of course, they are complex activities in themselves and include sub-activities such as requirements validation, architectural design, unit testing, etc. There are also supporting process activities such as documentation and software configuration management.

When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc., and the ordering of these activities. However, as well as activities, process descriptions may also include:

1. Products, which are the outcomes of a process activity. For example, the outcome of the activity of architectural design may be a model of the software architecture.

2. Roles, which reflect the responsibilities of the people involved in the process. Examples of roles are project manager, configuration manager, programmer, etc.

3. Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced. For example, before architectural design begins, a pre-condition may be that all requirements have been approved by the customer; after this activity is finished, a post-condition might be that the UML models describing the architecture have been reviewed.

Software processes are complex and, like all intellectual and creative processes, rely on people making decisions and judgments. There is no ideal process and most organizations have developed their own software development processes. Processes have evolved to take advantage of the capabilities of the people in an organization and the specific characteristics of the systems that are being developed. For some systems, such as critical systems, a very structured development process is required. For business systems, with rapidly changing requirements, a less formal, flexible process is likely to be more effective.

Sometimes, software processes are categorized as either plan-driven or agile processes. Plan-driven processes are processes where all of the process activities are planned in advance and progress is

measured against this plan. In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements. As experts discuss, each approach is suitable for different types of software. Generally, you need to find a balance between plan-driven and agile processes.

Although there is no 'ideal' software process, there is scope for improving the software process in many organizations. Processes may include outdated techniques or may not take advantage of the best practice in industrial software engineering. Indeed, many organizations still do not take advantage of software engineering methods in their software development.

Software processes can be improved by process standardization where the diversity in software processes across an organization is reduced. This leads to improved communication and a reduction in training time, and makes automated process support more economical. Standardization is also an important first step in introducing new software engineering methods and techniques and good software engineering practice.

Текст 3: Encapsulation

When developing complex systems, allowing indiscriminate access to an object's internals can be disastrous. It is all too easy for a careless, confused, or inept programmer to change an object's state in such a way as to corrupt the behavior of the entire system. A malicious programmer may intentionally tweak one or more objects to sabotage the system. In either case, if the software system controls a medical device or military missile system, the results can be deadly.

C++ provides several ways to protect the internals of an object from the outside world, but the simplest strategy is the one we have been using: fields and methods, generically referred to as class members, can be qualified as either public or private.

The compiler enforces the inaccessibility of private members. In simplerational.cpp list, for example, client code cannot directly modify the denominator instance variable of a Rational object making it zero. A client may influence the values of numerator and denominator only via methods provided by the class designer.

Accessibility rules, also called visibility rules or permissions, determine what parts of a class and/or object are accessible to the outside world. C++ provides a great deal of flexibility in assigning access permissions, but some general principles exist that, if followed, foster programs that are easier to build and extend.

• In general, fields should be private. Clients should not be able to arbitrarily change the state of an object. Allowing such might allow client code to put an object into an undefined state (for example, changing the denominator of a fraction to zero). An object's state should only change as a result of calling the object's methods.

The built-in primitive types like int and double offer no protection from client access. One exception to the private fields rule applies to simple objects that programmers naturally would treat as primitive types. Recall the geometric Point class found in mathpoints.cpp list. The x and y fields of a point object safely may assume any legitimate floating-point value, and it may be reasonable in some applications for clients to treat a Point object as a primitive type. In this case it is appropriate to make the x and y fields public.

• Methods that provide a service to client code should be part of the public section of the class. The author of the class must ensure that the public methods cannot place the object into an illegal state. For example, the method

```
void set_denominator(int d) {

denominator = d;

}
```

if part of the Rational class would permit client code to sabotage a valid fraction with a simple statement:

```
fract.set_denominator(0);
```

• Methods that assist the service methods but that are not meant to be used by the outside world should be in the private section of the class. This allows a public method to be decomposed into simpler,

perhaps more coherent activities without the threat of client code accessing these more primitive methods. These private methods are sometimes called helper methods or auxiliary methods.

Why would a programmer intentionally choose to limit access to parts of an object? Restricting access obviously limits the client's control over the objects it creates. While this may appear to be a disadvantage at first glance, this access restriction actually provides a number of advantages:

• Flexibility in implementation. A class conceptually can be separated into two parts:

– The class interface—the public part. Clients see and can use the public parts of an object. The public methods and public variables of a class constitute the interface of the class. A class's interface specifies what it does.

– The class implementation—the hidden part. Clients cannot see any private methods or private variables. Since this private information is invisible to clients, class developers are free to do whatever they want with the private parts of the class. A class's implementation specifies how it accomplishes what it needs to do.

We would like our objects to be black boxes: clients should not need to know how the objects work but merely rely on what objects can do.

Text 4: Microservices 101

When we think about building an application, we visualize it as a bunch of code glued together, running somewhere on the web for the purpose of providing people some value. It could be the homepage of your coffee shop, an e-commerce site or an amazing API which collects data from your watch.

Most of the time applications are basically that, a bunch of code, and that works perfectly for most common scenarios. They are effectively monoliths. A monolithic application is where all of the required logic is located within one unit (a war, a jar, a single application, one repository). Monolithic apps are considered an anti-pattern. They are responsible for all possible functionality:

- handling HTTP requests
- dealing with authentication
- executing domain logic
- database interactions
- communication with the user
- etc.

The problem is that, over time, the application evolves and its requirements become more complex. At some point, the monolithic application's architecture becomes a liability, where even the smallest change to the system requires re-building and re-deploying the entire application. In addition, the situation gets worse when you consider scaling: you have to run multiple instances of the monolith, even if you know that the bottleneck lies in a single component. This is where microservices architecture can be invaluable.

What is Microservices architecture?

Microservices is an approach to application architecture and development wherein an application is built in smaller, separate pieces. Each service is developed, tested and deployed independently. Although microservices architecture is fairly new, the basic concept behind it is one that will seem familiar to many software professionals. You could think of the Microservices Architectural Style as a specialization of SOA.

The only relation between different microservices is the data exchange accomplished through the APIs they expose. We can think of those microservices as built-in programs in a Unix-like OS, connected by pipes.

The benefits of microservices

- Faster and simpler deployments and rollbacks
- Elimination of long-term commitment to a single technology stack
- Improved fault isolation
- Independently scalable services
- Technology diversity
- Ability to write new features as plugins

The drawbacks of microservices

- Increased network communication
- Serialization between microservices
- Additional complexity in testing a distributed system
- Increased complexity in deployment

When should I switch from Monoliths to Microservices?

The complexity that should drive a switch to microservices can come from many sources: dealing with large teams, multi-tenancy, different business components evolving independently, and scaling. However, the most common driver is when a monolithic application becomes too large to modify and deploy.

You must fully understand your module boundaries and the domain's problem before attempting to use microservices; if you do not, you will end up with source code that is poorly written and hard to maintain.

Also, you must be ready to support:

- Rapid Provisioning: you should be able to fire up a new server in a matter of hours.
- Basic Monitoring: with many loosely-coupled services collaborating in production, things are bound to go wrong in ways that are difficult to detect in test environments.

- Rapid Application Deployment: with many services to manage, you need to be able

to quickly deploy them.

Текст 5 How to implement Microservices architecture?

The API gateway

Big enterprise front-ends might need to invoke large numbers of services, like Amazon does. Invoking requests often takes longer than receiving response data.

The API gateway handles incoming requests by sending subsequent requests to a number of microservices over a high-performance LAN, then returning an aggregated response to the consumer. The gateway should not contain any logic.

Keep in mind that remote calls can fail or hang without a response until a timeout occurs. There are a few techniques to deal with these issues. One of them, is the circuit-breaker pattern.

The basic idea behind the circuit breaker is very simple. You wrap a protected function call in a circuit breaker object, which monitors for failures. Once the failures reach a certain threshold, the circuit breaker trips, and all further calls to the circuit breaker return with an error, without the protected call being made at all.

Containers

Dealing with a number of microservices can easily become a very complex endeavor. Each can be written in a different programming language, with different dependencies. But most of these problems can be avoided by packaging each service as a separate container.

Containers are extremely fast to build and start. For example, Docker containers start much faster than a VM since only the application process starts rather than an entire OS.

Cross functional teams

When looking to split a large application into parts, often management focuses on the technology layer, leading to UI teams, server-side logic teams, and database teams. When teams are separated along these lines, even simple changes can lead to a cross-team project taking time, coordination and budgetary approval.

Microservices are best implemented by a single, cross-functional team, from the start (design) to finish (deployment and maintenance). Cross-functional teams give us the freedom to be agile, to innovate and to increase productivity by cutting out unnecessary layers. Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.

Who is using this architecture?

Large-scale web sites like Netflix and Amazon have evolved from a monolithic architecture to a microservices architecture.

Netflix is responsible for up to 30% of Internet traffic. It uses a large scale, service-oriented architecture to handle over a billion calls per day to their video streaming API from over 800 different device types.

Amazon originally had a two-tier architecture. In order to scale they migrated to a service-oriented architecture consisting of hundreds of backend services. Several applications call these services, including the applications that implement the amazon.com website and the web service API. For example, amazon.com calls 100-150 services to get the data it uses to build a single web page. Many other sites use a microservices approach including eBay, Soundcloud and Nike.

Microservices have existed for a long time, but their popularity has increased in recent years. Transitioning to microservices from a monolithic architecture is a slow process: it should be done incrementally because it requires a gradual change in the team structure and dynamics, besides the application's code.

Not every application would benefit from microservices. In addition, some applications are better off growing monolithically to determine which services will be useful to run independently. But don't fear the shift to microservices: if they are implemented thoughtfully, they might just save your application as it scales!

Text 6: How to Compare Databases

SQL Databases

Let us break SQL databases down into three basic groups: Traditional, MPP, columnar, and an emerging technology called NewSQL.

Traditional

These are the usual databases that we've seen for years. Some vendors might include MySQL, PostgreSQL, SQL Server (product), Sybase, Oracle Database, etc. They comply with SQL standards and provide full ACID transactions. Thus, you can do a lot of data transformation directly in the platform, and it is guaranteed that different people looking at the same data will see the same values. They are also relational in that data in different tables are typically joined together. Most often, these databases will run in a single node or a few nodes in a grid (some exceptions). These databases are generally used for business applications that process atomic (the lowest granularity) level transactions. They are also used when processing and storing data that involves something of physical value. For example, your company's ERP and Supply Chain systems will use these databases for their data storage. They are also pretty configurable to serve as good data warehousing solutions when tuned, configured, and modeled correctly.

MPP (Massively Paralleled Processing)

Generally speaking provide most of the same features as traditional databases but scale to hundreds or thousands of nodes. Vendors in this area include Oracle Exadata, Teradata, EMC Greenplum, and IBM Netezza. These are commonly used within large enterprises as the backbone of their data warehouses. Some vendors also provide specialized approaches to MPP databases. For example, platforms like Vertica Database, Actian Matrix, and Sybase IQ, provide column oriented storage in an MPP solution. SAP HANA and EXASOL are examples of MPP databases that operate primarily in memory.

NewSQL

Again, very similar to MPP and Traditional in that they are ACID compliant and support SQL standards, except that these solutions tend to rely very heavily or entirely on in-memory data storage. They also tend to scale horizontally. This is as opposed to the other solutions which primarily use disk as the

storage mechanism. The trade-off here is that NewSQL tends to store less data since disks can hold more data than RAM. In reality, even the MPP solutions tend to use a mixture of magnetic disk, solid state, and memory. These tend to be very popular for OLAP solutions as they can make data changes to large data sets very quickly. Examples of NewSQL are Oracle Timesten and VoltDB. In some cases, database vendors are blending these capabilities into their existing platform. Oracle Database12c allows you to "pin" tables to memory.

Columnar

Basically like it sounds, data is stored in columns. If you think about how data is typically stored in a traditional SQL DB, you have a single row of data with different kinds of data types. Just thinking about a user table, you may have an id, a first name, last name, address info, demographics, etc. When storing this data on a disk, it is hard to compress it in a consistent way. When data is stored in columns rather than rows, using the same user table example, all IDs are stored together, all names are stored together, etc. This allows the database to greatly compress data as now similar types of data are now stored together, resulting in less physical disk reads when querying large volumes of records.

The trade-off is that the more columns you select, the slower your query becomes as more data has to be read from the disk. This is in contrast to traditional SQL where adding more columns to a selected query typically has only a small impact on query performance. Besides, they are not well suited for use cases involving high concurrency of queries as a single query will use a large amount of available resources. They are also designed to be batch loaded and DML operations are very slow. This makes them very ill-suited for OLTP type workloads. Examples of columnar databases are MonetDB, Sybase IQ, and Actian Vector. Additionally, most MPP and other database vendors provide a mechanism for columnar compression.

NoSQL Datastores

There are so many of NoSQL datastores and it is really hard to classify them. Let us describe 5 main "types": Key-Value, Document, Graph, Elastic Search, and Time Series. The common denominator of NoSQL databases is that they are all designed to scale horizontally across multiple servers using an indexed and sharded key governed by a "gateway" node. Almost all NoSQL datastores are based on this concept and then apply some level of data structuring, indexing, or storage strategy to further provide specific advantages for some use cases.

They are generally used when you need to be able to read and write a lot of data concurrently. This is different from the traditional databases in that you can usually only tune those to be good at one or the other (again, in very broad terms). It is also important to note that NoSQL (in reference to the fact that these databases do not use SQL as their querying language) is not intended as a rejection of traditional SQL.

You have to keep in mind that without SQL compliance, most of these databases cannot transform data within their engine. You have to extract it somewhere else, write some code, and then put the transformed data wherever you need it. Largely, all NoSQL databases also lack ACID compliance and instead rely on "eventual consistency". At a high level, this means that different people querying the exact same data may see different values. You cannot join data together directly in these databases either.

Key-Value

These basically provide a very simple method of storing and retrieving binary data. For each record, you define its key and store a value. Usually, people will create large programming objects, serialize them, and then store it. Examples of these are Oracle (company) NoSQL & BerkeleyDB, Dynamo, and Riak. Cassandra can be put in this group too but in reality it provides more structure within each value than just an open binary store. This can be useful for storing something like a video game's user state, such as your Farmville farm (especially if that state is many megabytes+).

Document Store

These are similar to Key-Value stores except that instead of value just being a simple binary object, there is more structure behind the data. Most commonly, data is stored in a JSON format and a few of them use XML. Again, for each key, you will retrieve the document value. Documents tend to be very large objects and are usually accessed one at a time. Examples are MongoDB, CouchDB, and BaseX. These are useful for storing data with some structure such as legal documents or your entire online profile for a website.

Graph

Somewhat different than the Key-Value and Doc Store, these databases are designed to create "graphs" of data. This means you can define very complex relationships between different points of data. You can also easily traverse the data defined by relationships, even if it is many nodes away. Think about it as a social network, where you have friends, interests, check-ins, likes, etc., and you want to be able to define the relationships between these things and then query it. Examples include Neo4jandSPARQL.

Elastic Search

These types of NoSQL databases tend to get a lot of attention with misinformation as to how they can be used. In the most basic sense, these data stores index "words" and provide searching capabilities similar to typing something into Google. They are very good at finding things based on a key search over a very large distributed set of data, a kind of "enterprise search". There is an upfront cost to creating the index, and they are not designed to operations on large scans of data (i.e.: analytical workloads). Examples include Apache Solr, Elasticsearch, and Kibana.

Time Series

These data stores are based on the same basic principal of an indexed and sharded key. The difference, however, is that a timestamp is always part of a concatenated key. Records are appended using an "insert" operation only. The primary use case for this type of platform is for providing a persisted data store for data use cases where you will primary want to retrieve data as a trend. For example, a sensor with a unique ID that is constantly streaming data in. The data for any given unique ID is stored on the same node and sorted by its most recent timestamp. This makes it efficient for specific point in time or a time range retrieval of data for a specific ID. Many of these data stores also provide additional capabilities based on time series such as summary rollups, time based aggregations, and visual trending analysis. Examples of these include Druid, Influxdb, and OpenTSDB. I could also make a strong argument for putting HBase in this category.

Text 6: Software Engineering and the Web

The development of the World Wide Web has had a profound effect on all of our lives. Initially, the Web was primarily a universally accessible information store and it had little effect on software systems. These systems ran on local computers and were only accessible from within an organization. Around 2000, the Web started to evolve and more and more functionality was added to browsers. This meant that web-based systems could be developed where, instead of a special-purpose user interface, these systems could be accessed using a web browser. This led to the development of a vast range of new system products that delivered innovative services, accessed over the Web. These are often funded by adverts that are displayed on the user's screen and do not involve direct payment from users.

As well as these system products, the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational software. Instead of writing software and deploying it on users' PCs, the software was deployed on a web server. This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC. It also reduced costs, as user interface development is particularly expensive. Consequently, wherever it has been possible to do so, many businesses have moved to web-based interaction with company software systems.

The next stage in the development of web-based systems was the notion of web services. Web services are software components that deliver specific, useful functionality and which are accessed over the Web. Applications are constructed by integrating these web services, which may be provided by different companies. In principle, this linking can be dynamic so that an application may use different web services each time that it is executed.

In the last few years, the notion of 'software as a service' has been developed. It has been proposed that software will not normally run on local computers but will run on 'computing clouds' that are accessed over the Internet. If you use a service such as web-based mail, you are using a cloud-based system. A computing cloud is a huge number of linked computer systems that is shared by many users. Users do not buy software but pay according to how much the software is used or are given free access in return for watching adverts that are displayed on their screen.

The advent of the web, therefore, has led to a significant change in the way that business software is organized. Before the web, business applications were mostly monolithic, single programs running on single computers or computer clusters. Communications were local, within an organization. Now, software is highly distributed, sometimes across the world. Business applications are not programmed from scratch but involve extensive reuse of components and programs. This radical change in software organization has, obviously, led to changes in the ways that web-based systems are engineered. For example:

1. Software reuse has become the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.

2. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems should be developed and delivered incrementally.

3. User interfaces are constrained by the capabilities of web browsers. Although technologies such as AJAX mean that rich interfaces can be created within a web browser, these technologies are still

difficult to use. Web forms with local scripting are more commonly used. Application interfaces on web-based systems are often poorer than the specially designed user interfaces on PC system products.

The fundamental ideas of software engineering apply to web-based software in the same way that they apply to other types of software system. Experience gained with large system development in the 20th century is still relevant to web-based software.

Text 7: Software Engineering Ethics

Like other engineering disciplines, software engineering is carried out within a social and legal framework that limits the freedom of people working in that area. As a software engineer, you must accept that your job involves wider responsibilities than simply the application of technical skills. You must also behave in an ethical and morally responsible way if you are to be respected as a professional engineer.

It goes without saying that you should uphold normal standards of honesty and integrity. You should not use your skills and abilities to behave in a dishonest way or in a way that will bring disrepute to the software engineering profession. However, there are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

1. Confidentiality You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

2. Competence You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.

3. Intellectual property rights You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.

4. Computer misuse You should not use your technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses or other malware).

Professional societies and institutions have an important role to play in setting ethical standards. Organizations such as the ACM, the IEEE (Institute of Electrical and Electronic Engineers), and the British Computer Society publish a code of professional conduct or code of ethics. Members of these organizations undertake to follow that code when they sign up for membership. These codes of conduct are generally concerned with fundamental ethical behavior. Professional associations, notably the ACM and the IEEE, have cooperated to produce a joint code of ethics and professional practice. This code exists in both a short form and a longer form that adds detail and substance to the shorter version. The rationale behind this code is summarized in the first two paragraphs of the longer form:

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial

and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

In any situation where different people have different views and objectives you are likely to be faced with ethical dilemmas. For example, if you disagree, in principle, with the policies of more senior management in the company, how should you react? Clearly, this depends on the particular individuals and the nature of the disagreement. Is it best to argue a case for your position from within the organization or to resign in principle? If you feel that there are problems with a software project, when do you reveal these to management? If you discuss these while they are just a suspicion, you may be overreacting to a situation; if you leave it too late, it may be impossible to resolve the difficulties.

Such ethical dilemmas face all of us in our professional lives and, fortunately, in most cases they are either relatively minor or can be resolved without too much difficulty. Where they cannot be resolved, the engineer is faced with, perhaps, another problem. The principled action may be to resign from their job but this may well affect others such as their partner or their children.

Text 8: Software Pricing

In principle, the price of a software product to a customer is simply the cost of development plus profit for the developer. In practice, however, the relationship between the project cost and the price quoted to the customer is not usually so simple. When calculating a price, you should take broader organizational, economic, political, and business considerations into account, such as market opportunity, cost estimate uncertainty, contractual terms, requirements volatility, and financial health. You need to think about organizational concerns, the risks associated with the project, and the type of contract that will be used. These may cause the price to be adjusted upwards or downwards. Because of the organizational considerations involved, deciding on a project price should be a group activity involving marketing and sales staff, senior management, and project managers. To illustrate some of the project pricing issues, consider the following scenario:

A small software company, PharmaSoft, employs 10 software engineers. It has just finished a large project but only has contracts in place that require five development staff. However, it is bidding for a very large contract with a major pharmaceutical company that requires 30 person-years of effort over two years. The project will not start for at least 12 months but, if granted, it will transform the finances of the company. PharmaSoft gets an opportunity to bid on a project that requires six people and has to be completed in 10 months. The costs (including overheads of this project) are estimated at $1.2 million. However, to improve its competitive position, PharmaSoft decides to bid a price to the customer of $0.8 million.

This means that, although it loses money on this contract, it can retain specialist staff for the more profitable future projects that are likely to come on stream in a year's time.

As the cost of a project is only loosely related to the price quoted to a customer, 'pricing to win' is a commonly used strategy. Pricing to win means that a company has some idea of the price that the customer expects to pay and makes a bid for the contract based on the customer's expected price. This may seem unethical and unbusinesslike, but it does have advantages for both the customer and the system provider.

A project cost is agreed on the basis of an outline proposal. Negotiations then take place between client and customer to establish the detailed project specification. This specification is constrained by the agreed cost. The buyer and seller must agree on what is acceptable system functionality. The fixed factor in many projects is not the project requirements but the cost. The requirements may be changed so that the cost is not exceeded.

For example, say a company (OilSoft) is bidding for a contract to develop a fuel delivery system for an oil company that schedules deliveries of fuel to its service stations. There is no detailed requirements document for this system, so OilSoft estimates that a price of $900,000 is likely to be competitive and within the oil company's budget. After they are granted the contract, OilSoft then negotiates the detailed requirements of the system so that basic functionality is delivered. They then estimate the additional costs for other requirements. The oil company does not necessarily lose here because it has awarded the contract to a company that it can trust. The additional requirements may be funded from a future budget, so that the oil company's budgeting is not disrupted by a high initial software cost. This may involve renegotiating the project constraints and deliverables with the customer. A new schedule of when work should be completed also has to be established and agreed with the customer. If this renegotiation is unsuccessful or the risk mitigation actions are ineffective, then you should arrange for a formal project technical review. The objectives of this review are to find an alternative approach that will allow the project to continue, and to check whether the project and the goals of the customer and software developer are still aligned. The outcome of a review may be a decision to cancel a project. This may be a result of technical or managerial failings but, more often, is a consequence of external changes that affect the project. The development time for a large software project is often several years. During that time, the business objectives and priorities inevitably change. These changes may mean that the software is no longer required or that the original project requirements are inappropriate. Management may then decide to stop software development or to make major changes to the project to reflect the changes in the organizational objectives.

Text 9: The Planning Process

Project planning is an iterative process that starts when you create an initial project plan during the project startup phase. Plan changes are inevitable. As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule, and risk changes. Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be replanned.

At the beginning of a planning process, you should assess the constraints affecting the project. These constraints are the required delivery date, staff available, overall budget, available tools, and so on. In conjunction with this, you should also identify the project milestones and deliverables. Milestones are points in the schedule against which you can assess progress, for example, the handover of the system

for testing. Deliverables are work products that are delivered to the customer (e.g., a requirements document for the system).

The process then enters a loop. You draw up an estimated schedule for the project and the activities defined in the schedule are initiated or given permission to continue. After some time (usually about two to three weeks), you should review progress and note discrepancies from the planned schedule. Because initial estimates of project parameters are inevitably approximate, minor slippages are normal and you will have to make modifications to the original plan.

It is important to be realistic when you are creating a project plan. Problems of some description nearly always arise during a project, and these can lead to project delays. Your initial assumptions and scheduling should therefore be pessimistic rather than optimistic. There should be sufficient contingency built into your plan so that the project constraints and milestones don't need to be renegotiated every time you go around the planning loop.

If there are serious problems with the development work that are likely to lead to significant delays, you need to initiate risk mitigation actions to reduce the risks of project failure. In conjunction with these actions, you also have to replan the project. This may involve renegotiating the project constraints and deliverables with the customer. A new schedule of when work should be completed also has to be established and agreed with the customer.

If this renegotiation is unsuccessful or the risk mitigation actions are ineffective, then you should arrange for a formal project technical review. The objectives of this review are to find an alternative approach that will allow the project to continue, and to check whether the project and the goals of the customer and software developer are still aligned.

The outcome of a review may be a decision to cancel a project. This may be a result of technical or managerial failings but, more often, is a consequence of external changes that affect the project. The development time for a large software project is often several years. During that time, the business objectives and priorities inevitably change. These changes may mean that the software is no longer required or that the original project requirements are inappropriate. Management may then decide to stop software development or to make major changes to the project to reflect the changes in the organizational objectives.

Text 10: Embedded Systems Design

The design process for embedded systems is a systems engineering process in which the software designers have to consider in detail the design and performance of the system hardware. Part of the system design process may involve deciding which system capabilities are to be implemented in software and which in hardware. For many real-time systems embedded in consumer products, such as the systems in cell phones, the costs, and power consumption of the hardware are critical. Specific processors designed to support embedded systems may be used and, for some systems, special-purpose hardware may have to be designed and built.

This means that a top-down software design process, in which the design starts with an abstract model that is decomposed and developed in a series of stages, is impractical for most real-time systems. Low-level decisions on hardware, support software, and system timing must be considered early in the process. These limit the flexibility of system designers and may mean that additional software functionality, such as battery and power management, has to be included in the system.

Given that embedded systems are reactive systems that react to events in their environment, the most general approach to embedded, real-time software design is based on a stimulus-response model. A stimulus is an event occurring in the software system's environment that causes the system to react in some way; a response is a signal or message that is sent by the software to its environment.

You can define the behavior of a real-time system by listing the stimuli received by the system, the associated responses, and the time at which the response must be produced. (For example, stimuli and system responses for a burglar alarm system.)

Stimuli fall into two classes:

1. Periodic stimuli These occur at predictable time intervals. For example, the system may examine a sensor every 50 milliseconds and take action (respond) depending on that sensor value (the stimulus).

2. Aperiodic stimuli These occur irregularly and unpredictably and are usually signaled using the computer's interrupt mechanism. An example of such a stimulus would be an interrupt indicating that an I/O transfer was complete and that data was available in a buffer.

Stimuli come from sensors in the system's environment and responses are sent to actuators. A general design guideline for real-time systems is to have separate processes for each type of sensor and actuator. These actuators control equipment, such as a pump, which then makes changes to the system's environment. The actuators themselves may also generate stimuli. The stimuli from actuators often indicate that some problem has occurred, which must be handled by the system.

For each type of sensor, there may be a sensor management process that handles data collection from the sensors. Data processing processes compute the required responses for the stimuli received by the system. Actuator control processes are associated with each actuator and manage the operation of that actuator. This model allows data to be collected quickly from the sensor (before it is overwritten by the next input) and allows processing and the associated actuator response to be carried out later.

A real-time system has to respond to stimuli that occur at different times. You therefore have to organize the system architecture so that, as soon as a stimulus is received, control is transferred to the correct handler. This is impractical in sequential programs. Consequently, real-time software systems are normally designed as a set of concurrent, cooperating processes. To support the management of these processes, the execution platform on which the real-time system executes may include a real-time operating system. The functions provided by this operating system are accessed through the run-time support system for the real-time programming language that is used.

There is no standard embedded system design process. Rather, different processes are used that depend on the type of system, available hardware, and the organization that is developing the system. The following activities may be included in a real-time software design process: platform selection, stimuli/response identification, timing analysis, process design, algorithm design, data design, and process scheduling.

ЗЕТ-4. Тексты по специальности для письменного перевода с английского на русский язык со словарём.

Text 1: Application Architectures

Application systems are intended to meet a business or organizational need. All businesses have much in common—they need to hire people, issue invoices, keep accounts, and so on. Businesses operating in the same sector use common sector-specific applications. Therefore, as well as general business functions, all phone companies need systems to connect calls, manage their network, issue bills to customers, etc. Consequently, the application systems used by these businesses also have much in common.

These commonalities have led to the development of software architectures that describe the structure and organization of particular types of software systems. Application architectures encapsulate the principal characteristics of a class of systems. For example, in real-time systems, there might be generic architectural models of different system types, such as data collection systems or monitoring systems. Although instances of these systems differ in detail, the common architectural structure can be reused when developing new systems of the same type.

The application architecture may be re-implemented when developing new systems but, for many business systems, application reuse is possible without reimplementation. We see this in the growth of Enterprise Resource Planning (ERP) systems from companies such as SAP and Oracle, and vertical software packages (COTS) for specialized applications in different areas of business. In these systems, a generic system is configured and adapted to create a specific business application. For example, a system for supply chain management can be adapted for different types of suppliers, goods, and contractual arrangements.

As a software designer, you can use models of application architectures in a number of ways:

1. As a starting point for the architectural design process If you are unfamiliar with the type of application that you are developing, you can base your initial design on a generic application architecture.

2. As a design checklist If you have developed an architectural design for an application system, you can compare this with the generic application architecture. You can check that your design is consistent with the generic architecture.

3. As a way of organizing the work of the development team The application architectures identify stable structural features of the system architectures and in many cases, it is possible to develop these in parallel. You can assign work to group members to implement different components within the architecture.

4. As a means of assessing components for reuse If you have components you might be able to reuse, you can compare these with the generic structures to see whether there are comparable components in the application architecture.

5. As a vocabulary for talking about types of applications If you are discussing a specific application or trying to compare applications of the same types, then you can use the concepts identified in the generic architecture to talk about the applications.

There are many types of application system and, in some cases, they may seem to be very different. However, many of these superficially dissimilar applications actually have much in common, and thus can be represented by a single abstract application architecture. The following are the architectures of two types of application:

1. Transaction processing applications Transaction processing applications are database-centered applications that process user requests for information and update the information in a database. These are the most common type of interactive business systems. They are organized in such a way that user actions can't interfere with each other and the integrity of the database is maintained. This class of system includes interactive banking systems, e-commerce systems, information systems, and booking systems.

2. Language processing systems Language processing systems are systems in which the user's intentions are expressed in a formal language (such as Java). The language processing system processes this language into an internal format and then interprets this internal representation. The best-known language processing systems are compilers, which translate high-level language programs into machine code. However, language processing systems are also used to interpret command languages for databases and information systems, and markup languages such as XML.

Text 2: System Modeling

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. System modeling has generally come to mean representing the system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). However, it is also possible to develop formal (mathematical) models of a system, usually as a detailed system specification.

Models are used during the requirements engineering process to help derive the requirements for a system, during the design process to describe the system to engineers implementing the system and after implementation to document the system's structure and operation. You may develop models of both the existing system and the system to be developed:

1. Models of the existing system are used during requirements engineering. They help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system.

2. Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation. In a model-driven engineering process, it is possible to generate a complete or partial system implementation from the system model.

The most important aspect of a system model is that it leaves out detail. A model is an abstraction of the system being studied rather than an alternative representation of that system. Ideally, a representation of a system should maintain all the information about the entity being represented. An abstraction deliberately simplifies and picks out the most salient characteristics. For example, in the very unlikely event of this book being serialized in a newspaper, the presentation there would be an abstraction of the book's key points. If it were translated from English into Italian, this would be an alternative representation. The translator's intention would be to maintain all the information as it is presented in English.

You may develop different models to represent the system from different perspectives. For example:

1. An external perspective, where you model the context or environment of the system.

2. An interaction perspective where you model the interactions between a system and its environment or between the components of a system.

3. A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.

4. A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

These perspectives have much in common with Krutchen's 4 + 1 view of system architecture, where he suggests that you should document a system's architecture and organization from different perspectives.

A standard modeling language for object-oriented modeling is the UML, in which diagrams can be defined. The UML has many diagram types and so supports the creation of many different types of system model. However, some surveys showed that most users of the UML thought that five diagram types could represent the essentials of a system:

1. Activity diagrams, which show the activities involved in a process or in data processing.

2. Use case diagrams, which show the interactions between a system and its environment.

3. Sequence diagrams, which show interactions between actors and the system and between system components.

4. Class diagrams, which show the object classes in the system and the associations between these classes.

5. State diagrams, which show how the system reacts to internal and external events.

Let us see how these five key types of diagram are used in system modeling. When developing system models, you can often be flexible in the way that the graphical notation is used. You do not always need to stick rigidly to the details of a notation. The detail and rigor of a model depends on how you intend to use it. There are three ways in which graphical models are commonly used:

1. As a means of facilitating discussion about an existing or proposed system.

2. As a way of documenting an existing system.

3. As a detailed system description that can be used to generate a system implementation.

In the first case, the purpose of the model is to stimulate the discussion amongst the software engineers involved in developing the system. The models may be incomplete (so long as they cover the key points of the discussion) and they may use the modeling notation informally. This is how models are normally used in so-called 'agile modeling.' When models are used as documentation, they do not have to be complete as you may only wish to develop models for some parts of a system. However, these models have to be correct—they should use the notation correctly and be an accurate description of the system.

In the third case, where models are used as part of a model-based development process, the system models have to be both complete and correct. The reason for this is that they are used as a basis for generating the source code of the system. Therefore, you have to be very careful not to confuse similar symbols, such as stick and block arrowheads, that have different meanings.

Text 3: Social Networks

Do the names MySpace, Facebook, Orkut, etc. ring a bell? They probably do because they are some of the most popular sites on the Internet today. These sites are all called "social networking" sites because they help people meet and discuss things online. Each of these social networking sites has its own strengths: MySpace is especially popular among teenagers, Facebook is popular with college age people, Orkut is especially loved in Brazil, and CyWorld is the site to visit in South Korea. The common thread between all of these social networks is that they provide a place for people to interact, rather than a place to go to read or listen to "content".

Web 2.0

Social networks are considered to be web 2.0. What does this mean? To understand this, it is important to understand what the original web did (often called web 1.0).

Back in the nineties, the Internet – or Web – was a place to go to read articles, listen to music, get information, etc. Most people didn't contribute to the sites. They just "browsed" the sites and took advantage of the information or resources provided. Of course, some people did create their own sites. However, creating a site was difficult. You needed to know basic HTML coding (the original language the internet uses to "code" pages). It certainly wasn't something most people wanted to do as it could take hours to get a basic page just right. Things began to get easier when blogs (from web log) were introduced. With blogs, many more people began writing "posts", as well as commenting on other people's blogs.

MySpace Surprises Everybody

In 2003, a site named MySpace took the Internet by storm. It was trying to mimic the most popular features of Friendster, the first social networking site. It quickly became popular among young users and the rest was history. Soon everyone was trying to develop a social networking site. The sites didn't provide "content" to people, they helped people create, communicate and share what they loved including music, images and videos. They key to the success of these sites is that they provide a platform on which users create the content. This is very different from the beginning of the Internet, which focused on providing "content" for people to enjoy.

Key to Success

Relying on users to create content is the key to the success of web 2.0 companies. Beside the social networking sites discussed here, other huge success stories include: Wikipedia, Digg.com and the latest success – Twitter. All of these companies rely on the desire of users to communicate with each other, thereby creating the "content" that others want to consume.

Экзаменационные билеты для магистрантов по направлению «Программная инженерия» (ЕПИМ)

Билет №1

1. Topic “My Future Specialty.”

2. Annotate the text “Networks: LANs (Local Area Networks).”

Билет №2

1. Topic “My Future Specialty.”

2. Annotate the text “Faces of the Internet.”

Билет №3

1. Topic “My Future Specialty.”

2. Annotate the text “How Software Architects Drive Connected Vehicles.”

Билет №4

1. Topic “My Future Specialty.”

2. Annotate the text “New Vulnerability Discovered in Common Online Security.”

Билет №5

1. Topic “My Future Specialty.”

2. Annotate the text “Direct Client-to-Microservice Communication.”

Билет №6

1. Topic “My Future Specialty.”

2. Annotate the text “Using a Reactive Programming Model.”

Билет №7

1. Topic “My Future Specialty.”

2. Annotate the text “Organisations Not Taking Steps to Protect Smart Buildings.”

Билет №8

1. Topic “My Future Specialty.”

2. Annotate the text “How Smart Is Smart?”

Билет №9

1. Topic “My Future Specialty.”

2. Annotate the text “New City, New Risk.”

Билет №10

1. Topic “My Future Specialty.”

2. Annotate the text “Test Case.”

Билет №11

1. Topic “My Future Specialty.”

2. Annotate the text “Extreme Programming.”

Тексты для аннотирования

Text 1

Networks: LANs (Local Area Networks)

Networking allows two or more computer systems to exchange information and share resources and peripherals.

LANs are usually placed in the same building. They can be built with two main types of architecture: **peer-to-peer**, where two computers have the same capabilities, or **client server**, where one computer acts as the **server** containing the main hard disk and controlling the other **workstations** or **nodes**, all the devices linked in the network (e.g. printers, computers, etc.).

Computers in a LAN need to use the same **protocol**, or standard of communication. Ethernet is one of the most common protocols for LANs.

A **router**, a device that forwards data packets, is needed to link a LAN to another network, e.g. to the Net.

Most networks are linked with cables or wires but new **Wi-Fi**, **wireless fidelity**, technologies allow the creation of **WLAN**s, where cables or wire are replaced by radio waves.

To build a WLAN you need **access points**, radio-based receiver-transmitters that are connected to the wired LAN, and **wireless adapters** installed in your computer to link it to the network.

Hotspots are WLANs available for public use in places like airports and hotels, but sometimes the service is also available outdoors (e.g. university campuses, squares, etc.).

Text 2

Faces of the Internet

I. What the Internet Is

The **Internet** is an **Inter**national computer **Net**work made up of thousands of networks linked together. All these computers communicate with one another; they share data, resources, transfer information, etc. To do it they need to use the same language or **protocol**: **TCP/IP** (**Transmission Control Protocol/Internet Protocol**) and every computer is given an address or **IP number**. This number is a way to identify the computer on the Internet.

II. Getting Connected

To use the Internet you basically need a computer, the right connection software and a modem to connect your computer to a telephone line and then access your **ISP** (**Internet Service Provider**).

The **modem** (**mo**dulator-**dem**odulator) converts the digital signals stored in the computer into analogue signals that can be transmitted over **telephone lines**. There are two basic types: **external** with a cable that is plugged into the computer via a USB port, and **internal**, an expansion card inside the computer. A **PC card** modem is a different, more versatile option for laptops and mobile phones.

At first most computers used a **dial-up** telephone connection that worked through the standard telephone line. Now a **broadband** connection, a high data transmission rate Internet connection, has become more popular: either **ADSL** (**Asymmetric Digital Subscriber Line**), which allows you to use the same telephone line for voice and fast access to the Internet, or **cable**, offered by most TV cable providers.

The basic equipment has changed drastically in the last few years. You no longer need a computer to use the Internet. **Web TV** provides email and access to the Web via a normal TV set plus a high-speed modem. More recently, 3Generation mobile phones and PDAs, personal digital assistants, also allow you to go online with wireless connections, without cables.

Telephone lines are not essential either. **Satellites** orbiting the earth enable your computer to send and receive Internet files. Finally, the **power-line Internet**, is still under development, provides access via a power plug.

Text 3

How Software Architects Drive Connected Vehicles

Connecting vehicles to the Internet of Things is a fascinating field of activity for software architects. Introducing connectivity into vehicles pushes the door wide open for a broad spectrum of potential use cases. For example, consider receiving immediate warnings from other vehicles about traffic jams and black ice on the road ahead.

Or, you might want to remotely unlock your car doors with a smartphone to allow temporary access for family members. A mobile application lets the user issue control commands over a mobile Internet connection to a cloud-based connected-vehicle backend system. For example, the backend system might receive an “unlock doors” command and forward it to the connected car. After the car has received and executed the command, it reports the new door lock state back to the mobile device via the same communication path. However, designing a viable solution isn’t as straightforward as this process flow might suggest. The design must take into account a plethora of additional concerns, such as a car manufacturer’s various models, certain regional markets’ technological and legal peculiarities, and security aspects across all the involved systems. For example, the design must often consider secure communication channels and identity and access management. Owing to the large number and variety of those concerns, the architecture design process has an inherent complexity that requires management.

Text 4

New Vulnerability Discovered in Common Online Security

OpenSSL provides encryption protection for a range of applications on most types of computers and is similar to the encryption packages used by the web browsers Google Chrome (BoringSSL) and Firefox (Mozilla's Network Security Service (NSS)).

Dr. Yuval Yarom says he and colleagues have discovered that OpenSSL is vulnerable to a type of attack known as a “side channel attack.”

A side channel attack enables a hacker to take important information about software by examining the physical workings of a computer system – such as minute changes in power usage, or observing changes in timing when different software is being used.

Dr. Yarom has found that it is possible to “listen in” to the workings of the OpenSSL encryption software. His team measured highly sensitive changes in the computer's timing – down to less than one nanosecond. From these measurements, they recovered the private key which OpenSSL uses to identify the user or the computer.

“In the wrong hands, the private key can be used to ‘break’ the encryption and impersonate the user,” Dr Yarom says.

“At this stage we have only found this vulnerability in computers with Intel’s ‘Sandy Bridge’ processors. Computers with other Intel processors may not be affected in the same way.”

Dr Yarom says the likelihood of someone hacking a computer using this method is slim: “We seem to be the first to have done it, and under controlled conditions.”

“Servers, particularly Cloud servers, are a more likely target for this side-channel attack. It’s less likely that someone would use it against a home computer. There are so many easier-to-exploit vulnerabilities in home computers that it's unlikely someone would try to do this in the real world – but not impossible.”

Dr Yarom says there have been debates about this form of attack on OpenSSL for more than 10 years now, with some manufacturers claiming it couldn't be done. "But we have proven the vulnerability exists," he says.

"With OpenSSL being the most commonly used cryptographic software in the world right now, it's important for us to stay vigilant against any possible attack, no matter how small its chances might be."

"Once we discovered the vulnerability, we contacted the developers of OpenSSL and have been helping them to develop a fix for the problem," he says.

Text 4

The ZX Spectrum

In April 1982 a British company, headed by Sir Clive Sinclair, launched the ZX Spectrum computer on the market and sparked an IT revolution.

The tiny black computer with its rubber keys ignited the home computer age both in the UK and elsewhere, which led to an boom in computer manufacturing and developed software programmers whose talent is still evident today.

The ZX Spectrum was the brainchild of the entrepeneur Clive Sinclair, who had previously developed one of the first cheap and slim pocket calculators. The Spectrum was Sinclair's fourth computer, but was by far the most successful.

For many people, the ZX Spectrum was their first experience of using a computer and it soon gained a loyal following. In fact, it would not be a great exaggeration to credit Clive Sinclair and his ZX Spectrum with almost single-handedly creating the IT industry in the UK and providing the first learning tools for the programmers who shape today's video games and information technology.

Even today, there are programs being written for the Spectrum, though it has not been made for years. The computer was so successful that there are many nostalgic users all over the world, who look back on this machine with great affection.

Text 5

Direct Client-to-Microservice Communication

In theory, a client could make requests to each of the microservices directly. Each microservice would have a public endpoint (**https://*serviceName*.api.company.name**). This URL would map to the microservice's load balancer, which distributes requests across the available instances. To retrieve the product details, the mobile client would make requests to each of the services listed above.

Unfortunately, there are challenges and limitations with this option. One problem is the mismatch between the needs of the client and the fine-grained APIs exposed by each of the

microservices. The client in this example has to make seven separate requests. In more complex applications, it might have to make many more. For example, Amazon describes how hundreds of services are involved in rendering their product page. While a client could make that many requests over a LAN, it would probably be too inefficient over the public Internet and would definitely be impractical over a mobile network. This approach also makes the client code much more complex.

Another problem with the client directly calling the microservices is that some might use protocols that are not web-friendly. One service might use Thrift binary RPC while another service might use the AMQP messaging protocol. Neither protocol is particularly browser- or firewall-friendly and is best used internally. An application should use protocols such as HTTP and WebSocket outside of the firewall.

It is also difficult to refactor the microservices. Over time, we might want to change how the system is partitioned into services. For example, we might merge two services or split a service into two or more services. If, however, clients communicate directly with the services, then performing this kind of refactoring can be extremely difficult.

Because of these kinds of problems it rarely makes sense for clients to talk directly to microservices.

Text 6

Using a Reactive Programming Model

The API Gateway handles some requests by simply routing them to the appropriate backend service. It handles other requests by invoking multiple backend services and aggregating the results. With some requests, such as a product details request, the requests to backend services are independent of one another. In order to minimize response time, the API Gateway should perform independent requests concurrently. Sometimes, however, there are dependencies between requests. The API Gateway might first need to validate the request by calling an authentication service, before routing the request to a backend service. Similarly, to fetch information about the products in a customer's wish list, the API Gateway must first retrieve the customer's profile containing that information, and then retrieve the information for each product. Another interesting example of API composition is the Netflix Video Grid.

Writing API composition code using the traditional asynchronous callback approach quickly leads you to callback hell. The code will be tangled, difficult to understand, and error-prone. A much better approach is to write API Gateway code in a declarative style using a reactive approach. Examples of reactive abstractions include Future in Scala, CompletableFuture in Java 8, and Promise in JavaScript. There is also Reactive Extensions (also called Rx or ReactiveX), which was originally developed by Microsoft for the .NET platform. Netflix created RxJava for

the JVM specifically to use in their API Gateway. There is also RxJS for JavaScript, which runs in both the browser and Node.js. Using a reactive approach will enable you to write simple yet efficient API Gateway code.

Text 7

Organisations Not Taking Steps to Protect Smart Buildings

A survey conducted by the Electrical Contractors' Association (ECA) and Scottish electrical trade body Select found that half of the respondents note that hacking and its potential impact on unsecured networks is a potential barrier to installing connected technology.

Steve Martin, head of specialist groups at the ECA, said: "These figures are very concerning, particularly when you consider the inherent risks in the modern day of not securing your business from hackers. Clearly this is an area which clients urgently need to address, given the anticipated growth in smart installations over the coming years."

The survey collated responses from facilities managers, consultants, engineers, end clients and local authorities.

The ThinkFM conference held at the Science Museum in London was set to explore the ways in which technology is affecting facilities management.

BIFM's acting chief executive Linda Hausmanis said: "With new technology rapidly changing the way people interact with each other and the world around them, technology and its application is arguably becoming the most important resource for those working in facilities management."

Text 8

How Smart Is Smart?

The idea of smart cities relies on three fundamental ideas. Physical infrastructure can be used more efficiently as data analytics and artificial intelligence progress.

Engagement of the urban population with the city administration can be achieved through e-participation – or the carrying out of civic duties through the internet.

As technology continues to progress, computer systems will learn and adapt to challenges autonomously.

With proper implementation, smart cities will provide tremendous economic, social and cultural advantages for inhabitants, for instance through smart meters providing real-time data to the consumer and electricity company on consumption.

Indeed, smart cities are dependent on machine-to-machine (M2M) interactions and decision-making. This is in part a product of the sheer number of inputs and the frequency and speed with which associated calculations need to be completed. In the case of the energy grid, it would not be possible for a human operator to process all the data necessary to make decisions at the speed required by the system. But while M2M decision-making (M2MD) is an unavoidable and beneficial feature of smart cities, it is also one of the greatest risks.

The key challenge for organisations is ownership of cyber security more generally. By definition, smart cities will widen the number of stakeholders and risk-owners to whom private organisations will need to be accountable. Although various organisational functions have some form of responsibility for managing cyber security (IT, security, corporate risk, legal), it is imperative that there is clear senior ownership of and accountability for cyber security and that this is articulated in a clear organisational strategy.

Text 9

New City, New Risk

M2MD is a highly promising means of ensuring efficient automation across smart cities. But given the absence of human operators, the risk of a cascading error is significant.

For example, if a minor computing error caused a smart electricity reader to transmit inaccurate data readings to its control centre for a period of time, this could lead to an automated and mistaken assessment that a particular private organisation's premises required more electricity.

Smart cities will be composed of thousands if not millions of interconnected devices. Such a structure is a boon to criminals able to create or purchase and subsequently deploy self-propagating malware, variants of which have been known to proliferate across multiple connected networks.

Private sector organisations and municipal authorities will share ownership of systems and the responsibility for their security; beyond adding legal and financial costs for the private sector, this will create the need for highly complex pre-planned incident response schemes involving multiple parties.

The potential destructiveness of a cyber attack on smart cities is such that even the threat of compromise of the city's system is likely to be treated by governments and businesses as an existential one. In practice, private sector organisations are likely to be made accountable for the use of their networks by malicious intruders as the potential for the cascading effect of a single network breach will be magnified by smart cities.

Text 10

Test Case

A test case normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output, expected result, and actual result. Clinically defined a test case is an input and an expected result. This can be as pragmatic as 'for condition x your derived result is y', whereas other test cases described in more detail the input scenario and what results might be expected. It can occasionally be a series of steps (but often steps are contained in a separate test procedure that can be exercised against multiple test cases, as a matter of economy) but with one expected

result or expected outcome. The optional fields are a test case ID, test step, or order of execution number, related requirement(s), depth, test category, author, and check boxes for whether the test is automatable and has been automated. Larger test cases may also contain prerequisite states or steps, and descriptions. A test case should also contain a place for the actual result. These steps can be stored in a word processor document, spreadsheet, database, or other common repository. In a database system, you may also be able to see past test results, who generated the results, and what system configuration was used to generate those results. These past results would usually be stored in a separate table.

Text 11

Extreme Programming

Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent “releases” in short development cycles (timeboxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted. Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to “extreme” levels, on the theory that if some is good, more is better. It is unrelated to “cowboy coding”, which is more free-form and unplanned. It does not advocate “death march” work schedules, but instead working at a sustainable pace. Critics have noted several potential drawbacks, including problems with unstable requirements, no documented compromises of user conflicts, and a lack of an overall design specification or document.

Удалите жёлтое, это не надо. Это для меня, чтобы не создавать новый документ.

II. Network topology

Topology refers to the shape of a network. There are three basic physical topologies.

Star: there is a central device to which all the workstations are directly connected. This central position can be occupied by a server, or a hub, a connection point of the elements of a network that redistributes the data.

Bus: every workstation is connected to a main cable called a bus.

Ring: the workstations are connected to one another in a closed loop configuration.

There are also mixed topologies like the tree, a group of stars connected to a central bus.

III. WANs (Wide Area Networks)

WANs have no geographical limit and may connect computers or LANs on opposite sides of the world. They are usually linked through telephone lines, fibre-optic cables or satellites. The main transition paths within a WAN are high-speed lines called backbones.

Wireless WANs use mobile telephone networks.

The largest WAN in existence is the Internet.

III. Components of the Internet

The Internet consists of many systems that offer different facilities to users.

WWW, the World Wide Web, a collection of files or pages containing links to other documents on the Net. It's by far the most popular system. Most Internet services are now integrated on the Web.

Email, or electronic mail, for the exchange of messages and attached files.

Mailing lists (or listservs), based on programs that send messages on a certain topic to all the computers whose users have subscribed to the list.

Chat and instant messaging, for real-time conversations,; you type your messages on the keyboard.

Internet telephone, a system that lets people make voice calls via the Internet.

Video conference, a system that allows the transmission of video and audio signals in real time so the participants can exchange data, talk and see one another on the screen.

File Transfer Protocol (FTP), used to transfer files between computers.

Newsgroups, where people send, read and respond to public bulletin board messages stores on a central computer.

TELNET, a program that enables a computer to function as a terminal working from a remote computer and so use online databases or library catalogues.

Text 3

How Software Architects Drive Connected Vehicles

https://www.computer.org/cms/Computer.org/magazines/whats-new/2017/01/mso2016060041.pdf

Sören Frey, Lambros Charissis, and Jens Nahm, Daimler TSS

// There are two reasons why successful connected-vehicle projects often assign architecture-related responsibilities to individual experts acting as software architects. First, the experts help effectively manage complexity; second, they act as knowledge multipliers when development must scale up. //

Connecting vehicles to the Internet of Things is a fascinating field of activity for software architects. Introducing connectivity into vehicles pushes the door wide open for a broad spectrum of potential use cases.[1,2] For example, consider receiving immediate warnings from other vehicles about traffic jams and black ice on the road ahead.

Or, you might want to remotely unlock your car doors with a smartphone to allow temporary access for family members. Figure 1 depicts the high-level process flow for controlling the door lock state via mobile devices. A mobile application lets the user issue control commands over a mobile Internet connection to a cloud-based connected-vehicle backend system. For example, the backend system might receive an “unlock doors” command and forward it to the connected car. After the car has received and executed the command, it reports the new door lock state

https://www.sciencedaily.com/releases/2016/03/160302084557.htm

ЗЕТ-2. Темы для эссе:

1. Science and Technology – The Hazards of Today's Life
2. Science Is the First Step to Technological Progress
3. Science and New Challenges in the 21st Century
4. Business versus Software Engineering
5. How Do Emerging Technologies Affect Modern Society?
6. Learning and Doing: How to Build Your Development Way of Life?
7. Do Software Engineers Have Future in the New Energy Industry?
8. The Role of a Systems Engineer
9. Software Developers: The Brains behind Technology
10. The Vision of the Software Development

ЗЕТ-3. Темы для презентаций:

1. Famous Software Engineers
2. The Internet of Things for Smart Cities
3. How to Attract Audience to Your Product without an Advertising Budget?
4. How to Do Awesome Software Development?
5. Ethic Rules in An Agile Environment
6. What is the Use of the Touch Screen in Programming?
7. "Free" Software
8. Agile Development: A New Wave of Software Development
9. Reusable Software Development
10. How to Propel a Local Startup to the Global Market?

1 курс 2 семестр

ЗЕТ-1. Темы для мини-рефератов:

1. Keeping All Your Data in the Cloud – Is That Smart?
2. Recent Trends in Cloud Computing
3. Difference between Theft, Software Copyright Infringement, File Sharing and Piracy
4. Hash & Salt: How to Store?
5. Watch Out: Patent Trolls! Theory, Practice, and Current Examples
6. Entrepreneurship and Innovations in the IT Sphere
7. The Client Applications for Chat Servers
8. Development of Patent and License Accounting Information System

ЗЕТ-2. Темы для докладов:

1. Security and Privacy: Challenges for Cloud Computing
2. 5G Cellular Network: Architecture and Emerging Technologies
3. Security in the Client/Server Environment
4. The Internet of Battle Things
5. The Evolution of Software Architecture
6. Does Agile Make a Traditional Plan-Driven Project Management Approach Obsolete in a Software Engineering Environment?
7. SOA – Service-oriented Architecture
8. Defect Prediction

ЗЕТ-3. Темы для эссе:

1. Software and Data Systems in Everyday Life
2. Is BE+MBA a Better Saleable Combination Today?
3. Innovative Entrepreneurship in Modern Kyrgyzstan: A Dream or Reality?
4. New Skills for An Old Programmer?
5. Is There Only One “Proper” Way to Learn Programming?
6. Virtual Classrooms

ПРИЛОЖЕНИЕ 6

Шкала оценивания презентации

Общая оценка за презентацию учитывает 4 аспекта: раскрытие проблемы, представление презентации, её оформление, а также ответы на вопросы.

Раскрытие проблемы.

- **Минимальный ответ - 31-60 процентов.** Проблема не раскрыта. Отсутствуют выводы.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Проблема раскрыта не полностью. Выводы не сделаны или выводы не обоснованы.
- **Законченный полный ответ - 70-84 процента.** Проблема раскрыта. Проведен анализ проблемы без привлечения дополнительной литературы. Не все выводы сделаны или обоснованы.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Проблема раскрыта полностью. Проведен анализ проблемы с привлечением дополнительной литературы. Выводы сделаны.

Представление презентации.

- **Минимальный ответ - 31-60 процентов.** Представляемая информация логически не связана. Не использованы профессиональные термины.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Представляемая информация не систематизирована и не последовательна. Использован 1-2 профессиональных термина **.**
- **Законченный полный ответ - 70-84 процента.** Представляемая информация систематизирована и последовательна. Использовано более 2-х профессиональных терминов.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Представляемая информация систематизирована, последовательна и логически связана. Использовано более 5 профессиональных терминов.

Оформление

- **Минимальный ответ - 31-60 процентов.** Не использованы информационные технологии (PowerPoint). Больше 4 ошибок в представляемой информации
- **Изложенный, раскрытый ответ - 60-69 процентов.** Использованы информационные технологии (PowerPoint) частично. 3-4 ошибки в представляемой информации.
- **Законченный полный ответ - 70-84 процента.** Использованы информационные технологии (PowerPoint). Не более 2-х ошибок в представляемой информации
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Широко использованы информационные технологии (PowerPoint). Отсутствуют ошибки в представленной информации

Ответы на вопросы

- **Минимальный ответ - 31-60 процентов.** Нет ответов на вопросы **.**
- **Изложенный, раскрытый ответ - 60-69 процентов.** Только ответы на элементарные вопросы
- **Законченный полный ответ - 70-84 процента.** Ответы на вопросы полные или частично полные..
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Ответы на вопросы полные с приведением примеров и пояснений.

Шкала оценивания перевода

Перевод оценивается с учетом 4 аспектов: содержательной идентичности теста, логического и грамматического аспектов, а также соблюдения языковых норм и правил языка перевода, то есть стилистическая идентичность текста перевода

Содержательная идентичность текста перевода

- **Минимальный ответ - 31-60 процентов.** Неэквивалентная передача смысла: ошибки представляют собой искажение содержания оригинала.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Неточность передачи смысла: ошибки приводят к неточной передаче смысла оригинала, но не искажают его полностью.
- **Законченный полный ответ - 70-84 процента.** Погрешности перевода: погрешности перевода не нарушают общего смысла оригинала.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Эквивалентный перевод: содержательная идентичность текста перевода

Лексические аспекты перевода

- **Минимальный ответ - 31-60 процентов.** Использование эквивалентов для перевода 40-50 % текста.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Использование эквивалентов для перевода 60-70% текста.
- **Законченный полный ответ - 70-84 процента.** Использование эквивалентов для 80-90% текста.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Использование эквивалентов для перевода 100% текста.

Грамматические аспекты перевода

- **Минимальный ответ - 31-60 процентов.** Использование грамматических эквивалентов для 40-50 % текста.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Использование грамматических эквивалентов для 60-70% текста.
- **Законченный полный ответ - 70-84 процента.** Погрешности в переводе основных грамматических конструкций, характерных для газетного стиля речи.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Эквивалентный перевод с использованием основных грамматических конструкций, характерных для газетного стиля речи и художественного произведения.

Соблюдение языковых норм и правил языка перевода: стилистическая идентичность текста перевода.

- **Минимальный ответ - 31-60 процентов.** Соблюдение языковых норм и правил языка перевода 40-50% текста.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Соблюдение языковых норм и правил языка перевода 60-70% текста.
- **Законченный полный ответ - 70-84 процента.** Соблюдение языковых норм и правил языка перевода 80-90% текста.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Соблюдение языковых норм и правил языка перевода газетного и художественного текста.

Шкала оценивания сочинения, эссе

При оценивании сочинения учитываются 5 аспектов: решение коммуникативной задачи, то есть содержание сочинения, логика и организация текста, лексика, грамматика, орфография и пунктуация.

Решение коммуникативной задачи (содержание)

- **Минимальный ответ - 31-60 процентов.** Задание выполнено частично: содержание слабо отражает те аспекты, которые указаны в задании; много нарушений стилевого оформления; почти не соблюдаются принятые в языке нормы вежливости.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Задание выполнено не полностью: содержание не отражает все аспекты, указанные в задании; часто встречаются нарушения стилевого оформления; в основном не соблюдаются принятые в языке нормы вежливости.
- **Законченный полный ответ - 70-84 процента.** Задание выполнено: некоторые аспекты, указанные в задании раскрыты не полностью; имеются отдельные нарушения стилевого оформления речи; в основном соблюдены принятые в языке нормы вежливости.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Задание выполнено полностью: содержание отражает все аспекты, указанные в задании, стилевое оформление речи выбрано правильно с учетом цели высказывания и адресата; соблюдены принятые в языке норм вежливости.

Логика и организация текста

- **Минимальный ответ - 31-60 процентов.** Логика высказываний нарушена почти на всем протяжении сочинения.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Высказывание не всегда логично; имеются недостатки или ошибки в использовании средств логической связи, их выбор ограничен; деление текста на абзацы нелогично или отсутствует; имеются многочисленные ошибки в формате высказывания..
- **Законченный полный ответ - 70-84 процента.** Высказывание в основном логично; имеются отдельные недостатки при использовании средств логической связи; имеются отдельные недостатки при делении текста на абзацы; имеются отдельные нарушения формата высказывания.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Высказывание логично; средства логической связи выбраны правильно; текст разделен на абзацы; формат высказывания выбран правильно.

Лексика

- **Минимальный ответ - 31-60 процентов.** Бедный словарный запас. Слова не сочетаются друг с другом. Тест понятен фрагментарно.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Использован неоправданно ограниченный словарный запас; часто встречаются нарушения в использовании лексики, некоторые из которых могут затруднять понимание текста.
- **Законченный полный ответ - 70-84 процента.** Используемый словарный запас соответствует поставленной задаче, однако встречаются отдельные неточности в употреблении слов либо словарный запас ограничен. Но лексика использована правильно (3-7 ошибок).
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Используемый словарный запас соответствует поставленной задаче; практически нет нарушений в использовании лексики (1-2 ошибки).

Грамматика

- **Минимальный ответ - 31-60 процентов.** Грамматические правила серьезно нарушаются (более 10 ошибок).
- **Изложенный, раскрытый ответ - 60-69 процентов.** Либо часто встречаются ошибки элементарного уровня, либо ошибки немногочисленны, но затрудняют понимание текста.
- **Законченный полный ответ - 70-84 процента.** Имеется ряд грамматических ошибок, не затрудняющих понимание текста (3-7 ошибок).
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Используются грамматические структуры в соответствии с поставленной задачей. Практически отсутствуют ошибки. (1-2 ошибки).

Орфография и пунктуация

- **Минимальный ответ - 31-60 процентов.** Имеется ряд орфографических или пунктуационных ошибок, которые значительно затрудняют понимание текста .
- **Изложенный, раскрытый ответ - 60-69 процентов.** Наблюдаются орфографические и пунктуационные ошибки, которые иногда затрудняют понимание текста.
- **Законченный полный ответ - 70-84 процента.** Имеется ряд орфографических и пунктуационных ошибок, не затрудняющих понимание текста (3-7 ошибок).
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Орфографические ошибки практически отсутствуют. Текст разделен на предложения с правильным пунктуационным оформлением.

Шкала оценивания доклада, сообщения

Общая оценка за доклад учитывает 4 аспекта: раскрытие проблемы, представление текста доклада, его оформление, а также ответы на вопросы.

Раскрытие проблемы.

- **Минимальный ответ - 31-60 процентов.** Проблема не раскрыта. Отсутствуют выводы.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Проблема раскрыта не полностью. Выводы не сделаны или выводы не обоснованы.
- **Законченный полный ответ - 70-84 процента.** Проблема раскрыта. Проведен анализ проблемы без привлечения дополнительной литературы. Не все выводы сделаны или обоснованы.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Проблема раскрыта полностью. Проведен анализ проблемы с привлечением дополнительной литературы. Выводы сделаны.

Представление доклада.

- **Минимальный ответ - 31-60 процентов.** Представляемая информация логически не связана. Не использованы профессиональные термины.
- **Изложенный, раскрытый ответ - 60-69 процентов.** Представляемая информация не систематизирована и не последовательна. Использован 1-2 профессиональных термина **.**
- **Законченный полный ответ - 70-84 процента.** Представляемая информация систематизирована и последовательна. Использовано более 2-х профессиональных терминов.
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Представляемая информация систематизирована, последовательна и логически связана. Использовано более 5 профессиональных терминов.

Оформление

- **Минимальный ответ - 31-60 процентов.** Не использованы информационные технологии (PowerPoint). Больше 4 ошибок в представляемой информации
- **Изложенный, раскрытый ответ - 60-69 процентов.** Использованы информационные технологии (PowerPoint) частично. 3-4 ошибки в представляемой информации.
- **Законченный полный ответ - 70-84 процента.** Использованы информационные технологии (PowerPoint). Не более 2-х ошибок в представляемой информации
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Широко использованы информационные технологии (PowerPoint). Отсутствуют ошибки в представленной информации

Ответы на вопросы

- **Минимальный ответ - 31-60 процентов.** Нет ответов на вопросы **.**
- **Изложенный, раскрытый ответ - 60-69 процентов.** Только ответы на элементарные вопросы
- **Законченный полный ответ - 70-84 процента.** Ответы на вопросы полные или частично полные..
- **Образцовый, примерный, достойный подражания ответ - 85-100 процентов.** Ответы на вопросы полные с приведением примеров и пояснений.

ТЕХНОЛОГИЧЕСКАЯ КАРТА ПО СРС

Дисциплины: **Иностранный язык**
Направление:
Курс: **1, семестр 1, кредитов3 , зачет с оценкой**
Кафедра: **Иностранных языков**
Преподаватели:

В процессе освоения дисциплины студенты должны пройти три контрольные точки.

	Виды контроля/ формы контроля		**Набираемые баллы**		**График контроля недели (дата)**
			Зачетный минимум	**Зачетный максимум**	
Knowledge Management	Модуль 1	Текущий контроль: Пересказ и аннотирование текстов, устные сообщения и дискуссии, грамматические и лексические упражнения, внеаудиторное чтение, посещаемость и активность.	7	13	
		Защита мини-реферата по предложенным темам.	6	10	
Perspectives on Science and Technology	Модуль 2	Текущий контроль: Пересказ и аннотирование текстов, устные сообщения и дискуссии, грамматические и лексические упражнения, внеаудиторное чтение, посещаемость и активность.	7	13	
		Защита докладов по заданным темам	6	10	
Science in Everyday Life	Модуль 3	Текущий контроль: Пересказ и аннотирование текстов, устные сообщения и дискуссии, грамматические и лексические упражнения, внеаудиторное чтение, посещаемость и активность.	8	14	
		Написание эссе на тему «Software and Data Systems in Everyday Life».	6	10	
	Всего за семестр		40	70	
	Промежуточный контроль (зачет с оценкой)		20	30	
	Итоговый балл за семестр		60	100	