МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

МАГИСТРАТУРА

УТВЕРЖДАЮ
Декан факультета АВТ
Рева И.Л.

1. Индивидуальный план работы студента

___Ветчинниковой Алины Евгеньевны_____

(Фамилия, Имя, Отчество)

2. Факультет_автоматики и вычислительной техники_____
3. Кафедра ___вычислительной техники_____
4. Научный руководитель__Якименко Александр Александрович_____
5. Период обучения в магистратуре ____2020/2022 гг._____
6. Направление подготовки _09.04.04 – Программная инженерия
7. Наименование профессионально-образовательной программы (специализации) Разработка программного обеспечения информационных систем
8. Тема магистерской диссертации и её аннотация_____

Разработка и исследование алгоритма фильтрации pm-файлов в условиях высоконагруженного трафика_____

9. Срок представления студентом диссертации _____май 2022г._____

10. Содержание научно-исследовательской части программы

Семестр 1

№ п.п.	Наименование этапов, позиций	Дата	Приложение	Балл	Подпись руковод. о выполнении
1.	Индивидуальное задание в рамках магистерского исследования	18 неделя	Отчет	45	
2.	Посещение лекции от Huawei	7.10.20	Скрин	5	
3.	Посещение Huawei Honor Cup 2020	15.10.20	Фото	5	
4.	Зачетное выступление на научном семинаре	16.12.20	Презентация	10	
5.	Прохождение факультативного курса «Киберфизические системы: теория и приложения»	30.11.20 – 31.01.21		15	
	Итоговая аттестация		зачет	80	

Подпись студента _____

(дата)

Подпись научного руководителя _____

Подпись зав.кафедрой _____

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Факультет автоматики и вычислительной техники
Кафедра вычислительной техники

Индивидуальная работа
по теме магистерской диссертации
«Разработка и исследование алгоритма фильтрации pm-файлов в условиях высоконагруженного трафика»

Факультет: АВТФ
Группа: АПИМ-20
Студент: Ветчинникова А.Е.
Научный руководитель: Якименко А.А.

НОВОСИБИРСК
2020

В ходе первого семестра обучения и выполнения магистерской диссертации были выполнены следующие действия:

- Сформированы первоначальная цель и задачи;
- Проведен аналитический обзор источников;
- Скорректированы цель и задачи.

Первоначальная цель и задачи

Цель: Разработать алгоритм фильтрации csv-файлов с возможностью изменения правил фильтрации без выпуска новой версии приложения.

Актуальность: Современные информационные системы, как правило являются высоконагруженными, соответственно встает задача быстрой и беспрерывной обработки данных.

Задачи:

- Провести обзор существующих технологий динамической интерпретации кода
- Разработать алгоритм фильтрации pm-файлов с использованием технологи динамической интерпретации кода
- Провести тестирование

Аналитический обзор

1. Сбор материала с сохранением источников (ссылок) и его первичная классификация

Таблица 1. Первичная классификация собранного материала

	Ссылка	Название сайта и раздела	Аннотация
1	https://www.sciencedirect.com/science/article/pii/S1474667015404689	Sciencedirect. Towards Frame-based Source Code Generation for Heterogeneous Real-time Environments	Генерации исходного кода, задач и шаблонов взаимодействия для различных операционных систем реального времени.
2	https://habr.com/ru/company/dsec/blog/433034/	Habr. SpEL injection	Особенности использования SpEL, уязвимости, область применения и способы обнаружения инъекций.
3	https://www.oreilly.com/library/view/oracle-database-programming/9781555583293/ch05.html	Oreilly. 5 - Database Scripting using Non-Java Languages	Использование JACL, Kawa и Groovy для работы с базами данных.
4	https://www.researchgate.net/publication/325503106_SQL_Injection_Attack_classification_through_the_feature_extraction_of_SQL_query_strings_using_a_Gap-Weighted_String_Subsequence_Kernel	ResearchGate. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel	Тестирование приложения на способность ввода SQL инъекций, способы внедрения и защиты от них, обработка данных для обеспечения безопасности.
5	https://habr.com/r	Habr. Динамическая	Рассматривается структура

	u/post/154891/	интерпретация метамоделей.	компонентов ИС с динамической интерпретацией метамоделей
6	https://www.sciencedirect.com/science/article/pii/S0924271617303660	Sciencedirect. A framework for capturing, statistically modeling and analyzing the evolution of software models	Новая метод. основа для сбора и статистического моделирования развития моделей при разработке программного обеспечения.
7	https://www.researchgate.net/publication/221000769_Incremental_concrete_syntax_for_embedded_languages	ResearchGate. Incremental concrete syntax for embedded languages with support for separate compilation.	Представлен подход, который значительно сокращает необходимые усилия для реализации встроенного шаблона DSL с конкретным синтаксисом.
8	https://habr.com/ru/post/227753/	Habr. Метапрограммирование с примерами на JavaScript	Основы метапрограммирования, приемы и преимущества. Пример на JS.
9	https://www.semanticscholar.org/paper/Annotated-regular-expressions-and-input-driven-Sakharov/972b297765e1f62ce465c64e24ea0b4690b3cfad#extracted	Semantic Scholar. Annotated regular expressions and input-driven languages	Использование регулярных выражений.
10	https://www.scitepress.org/Papers/2017/63558/63558.pdf	Scitepress. Efficient runtime metaprogramming services for Java	Метод добавления функций метапрограммирования в Java, для повышения адаптивности во время выполнения, с использованием надежности системы статических типов и производительности

			виртуальной машины.
11	https://hal.archives-ouvertes.fr/inria-00429568/	HAL archives-ouvertes. Flexible metaprogramming and AOP in Java	Рассматривается метапрограммирование, рефлексия и аспектно-ориентированное программирование.
12	https://www.sciencedirect.com/science/article/abs/pii/S1477842408000146	Sciencedirect. Zero—a blend of static typing and dynamic metaprogramming	Экспериментальный статически типизированный, полностью объектно-ориентированный язык программирования с рефлексией.
13	https://www.semanticscholar.org/paper/Metaprogramming-Applied-to-Web-Component-Deployment-Löwe-Noga/64295805eb36e8bd5378aa2ce5b7ea38c732e99c	Semantic Scholar. Metaprogramming Applied to Web Component Deployment	Пример метапрограммы на 2500 строк для развертывания web-компонентов с добавлением специальных XML-конфигураций.
14	https://www.sciencedirect.com/science/article/pii/S0743106694900 8	Sciencedirect. Hierarchical representation of legal knowledge with metaprogramming in logic.	Приложение, в котором, в отличие от большинства метапрограмм, метапрограммирование не связано с управлением логикой выполнения программы.
15	https://habr.com/ru/company/haulmont/blog/445114/	Habr. Улучшенный sandboxing для Groovy скриптов	Механизмы интеграции скриптов на Groovy в Java код.
16	https://www.codeflow.site/ru/article/freemarker-in-spring-mvc-tutorial	Codeflow. Введение в использование FreeMarker в Spring MVC	Обзор FreeMarker и примеры использования в Spring MVC.

17	https://habr.com/ru/post/420549/	Habr. FreeMarker шаблоны	Обзор Apache FreeMarker
18	https://community.jaspersoft.com/wiki/using-groovy-filter-expression	Jaspersoft. Using Groovy in a Filter Expression	Пример настройки SQLGenerator для подстановки выражений Groovy
19	https://habr.com/ru/post/137446/	Habr. Метапрограммирование	Метапрограммирование – обзор, классификация, задачи.
20	https://dzone.com/articles/learn-spring-expression-language-with-examples	DZone. Learning the Spring Expression Language	Введение в SpEL, синтаксис и особенности языкк, основные компоненты, примеры использования.

2. Краткое изложение (цитирование) содержания источников, имеющего отношение к проблеме.

1. Towards Frame-based Source Code Generation for Heterogeneous Real-time Environments

   Основанное на фреймах создание задач и шаблонов связи в гетерогенных распределенных средах реального времени предлагает много новых возможностей для процесса разработки программного обеспечения в таких системах. Ошибки во время программирования можно уменьшить за счет абстрагирования нефункциональных шаблонов и их предоставления в менее сложной среде. Эти методы также подразумевают минимизацию требований к уровню знаний разработчика. Благодаря реализации на C / C++. HyAppLang можно напрямую использовать для разработки систем, основанных на больших задачах, с учетом взаимосвязей связи. Эти задачи могут быть распределены по разным операционным системам, в то время как разработчик приложения должен знать только свойства инфраструктуры HyROS и аннотации HyAppLang для создания и использования компонентов задач и связи.

2. SpEL injection

В Spring SpEL используется сплошь и рядом. Показательным примером можно считать Spring Security, где права назначаются с помощью SpEL выражения, Apache Camel использует SpEL API для формирования header'ов. Главной уязвимостью является **CVE 2018-1273 Spring Data Commons**, которая нашлась в методе setPropertyValue и основывалась на двух проблемах:

1) Недостаточная санитизация значений переменной, попадающей в ExpressionParser.

2) Исполнение выражения в рамка стандартного контекста.

3. 5 - Database Scripting using Non-Java Languages

Рассмотрена работа с базами данных с использованием скриптовых языков, отличных от Java. Oracle предлагает только PL / SQL и Java для программирования баз данных; однако с появлением компиляторов Java для других языков, появляется возможность повторно использовать виртуальную машину Java для выполнения байт-кода, полученного в результате компиляции в базе данных языков, отличных от Java. Рассматривается использование JACL, Kawa и Groovy для работы с БД. Преимущество Groovy заключается в том, что в отличие от других скриптовых языков он добавляет высокоуровневое сопоставление с существующими библиотеками Java в примерах SQL и XML.

4. SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel

В данной работе исследуется уязвимость программного продукта к SQL-инъекциям. Проводится тестирование – сначала определяются все параметры URL и столбцов/полей доступных для пользовательского ввода, а затем происходит их группировка по типу SQL запроса, производится инъекция. Авторы обнаружили, что параметры URL-адреса и входные столбцы или поле с текстовым типом уязвимы для SQL-инъекции. Уязвимость системы вызвана вводом и содержанием

параметров URL-адреса, которые явно запрашиваются в базе данных без какой-либо проверки, фильтрации или обработка в первую очередь, поэтому реализуется решение для обработки с использованием методов регулярного выражения, оператора подготовки и управляющей строки MySQL. Усовершенствованная прикладная система с этими тремя решениями для обработки проверяется путем повторного тестирования, как и в предыдущем тесте, и результаты системы являются безопасными и не могут быть использованы с помощью SQL-инъекции, поэтому можно сделать вывод, что проверенные решения для обработки эффективно предотвращают наложение системы путем внедрения SQL.

5. Динамическая интерпретация метамоделей

Основные классы задач, которые требуют повышения уровня абстракции ПО:

a. Задачи с **динамической предметной областью**, где изменение структуры и параметров модели являются штатным режимом функционирования
b. Задачи включающие **слабо-связанные данные** или данные с непостоянной структурой, параметрами или логикой обработки

Многослойная структура компонентов ИС с динамической интерпретацией метамоделей замыкается снизу на самоописывающее хранилище (ограниченное уровнем абстракции метамодели), а сверху замыкается на пользователя, который может модифицировать не только данные, но и метаданные, переналаживая ИС под постоянно меняющиеся требования решаемой задачи, в этом и состоит гибкость подхода и упрощение модификации систем, снижение необходимого уровня квалификации пользователя при внесении изменений в структуру и функции. Место же архитектора программных решений и программистов в создании программного обеспечения с повышенным уровнем абстракции, т.е. в реализации метамодели (что является, естественно, более трудной инженерной задачей). Но в конечном итоге, подход окупает

себя, благодаря повышению повторного использования кода с высокой абстракцией, автоматизации многих задач связывания компонентов, снижению влияния человеческого фактора при модификации и интеграции систем и упрощении интеграции между компонентами программных систем.

6. A framework for capturing, statistically modeling and analyzing the evolution of software models

   В данной статье представлена новая методологическая основа для сбора и статистического моделирования развития моделей при разработке программного обеспечения. Платформа фиксирует изменения между версиями моделей с точки зрения операций редактирования как низкого (внутреннего), так и высокого уровня (видимые разработчику), применяемых между версиями. Основное применение разработанной платформы состоит в том, чтобы контролировать создание реалистичных историй моделей, которые предназначены для использования инструментов управления версиями моделей. Авторы представляют архитектуру генератора моделей и показывают, как генерировать случайные последовательности из статистических моделей, которые управляют процессом генерации. Дальнейшее использование статистических моделей включает различные задачи прогнозирования и моделирования.

7. Incremental concrete syntax for embedded languages with support for separate compilation

   В данной статье представлен новый подход для включения необходимого синтаксиса для встроенных DSL. Основная идея заключается в том, чтобы дать возможность разработчикам языков постепенно определять синтаксис DSL с помощью отдельных грамматик. По сравнению с аналогичными работами, основное преимущество данного подхода состоит в том, что необходимо указывать только части конкретного синтаксиса DSL. Это имеет четыре основных приемущества:

1) необходимость указать конкретный синтаксис DSL для тех частей, где конкретный синтаксис языка хоста конфликтует с указанием данного синтаксиса домена, 2) такой подход поддерживает постепенное определение конкретного синтаксиса DSL, что помогает сократить начальные усилия, необходимые при разработке EDS, 3) реализация встроенного DSL может быть независимой от языка хоста. Это облегчает повторное использование EDSLimplementations, если язык хоста компилируется в представление байт-кода, такое как байт-код Java или CIL,

4) разработчики могут отдельно компилировать реализации EDSL и распространять основной байт-код. Позже конечные пользователи могут загружать двоичные версии EDSL и составной файл

8. Метапрограммирование с примерами на JavaScript

Приемы метапрограммирования:

- Стиль описания задачи: декларативный (метаданные), использование императивных и функциональных вставок
- Хеши (ассоциативные массивы) заранее не знаем ключ: let a = {}; a[key] = value;
- Интерпретация строк, придумываем свои синтаксисы или берем общепринятые (json, js, regexp ...) ;
- Примеси (mixins): заранее не знаем куда добавим function mixin(a) { a.fn= () => {… } };
- Замыкания (closures): персонализируем функции fn = (a => () => a * 2)(value)

Преимущества метапрограммирования:

- Размер кода: чаще резко уменьшается, но иногда может немного увеличиваться;
- Быстродействие: незначительно снижаться, но при грамотной реализации остается примерно тем же;
- Гибкость: программный код становится более универсальным, сфера применения ПО расширяется;
- Интеграция: обычно значительно упрощается и требует меньше изменений кода;

- Удовольствие от работы: метапрограммировать интереснее поэтому удовольствия и мотивации больше;
- Скорость разработки: разрабатывать дольше, а поддерживать значительно проще, экономится уйма времени

9. Annotated regular expressions and input-driven languages

Регулярные выражения расширяются путем разделения терма на левые скобки, правые скобки и нейтральные термы, а также путем добавления аннотаций термов. Эти расширенные выражения определяют набор хорошо вложенных языков, управляемых вводом. Они анализируются в линейном времени по размеру входных данных. Деревья синтаксического анализа основаны на скобках и приоритетах операторов.

10. Efficient runtime metaprogramming services for Java

Авторы предлагают метод добавления функций метапрограммирования в Java, для повышения адаптивности во время выполнения, с использованием надежности системы статических типов и производительности виртуальной машины. Данный метод поддерживает динамическое добавление, удаление и замену методов и полей класса, а также динамическую генерацию кода. Возможности метапрограммирования предоставляются в виде библиотеки, поэтому ни язык Java, ни его виртуальная машина не изменяются. Авторы оценивают разработанную систему под названием JMPLib и сравнивают ее с существующими системами метапрограммирования для платформы Java и других высоко оптимизированных динамических языков. JMPLib обеспечивает такую же производительность во время выполнения, что и существующая самая быстрая система, которая изменяет реализацию виртуальной машины Java. Более того, данная система не снижает производительность, когда метапрограммирование не используется, и потребляет меньше ресурсов памяти, чем остальные реализации для платформы Java.

11. Flexible metaprogramming and AOP in Java

Основываясь на единой модели частичного отражения, Reflex предоставляет как структурные, так и поведенческие возможности, которые хорошо подходят для поддержки аспектно-ориентированного программирования. Будучи универсальным ядром для АОП, Reflex дополнительно поддерживает композицию аспектов, а также определение (зависящих от предметной области) языков аспектов. В результате Reflex позволяет создавать приложения, объединяя аспекты, написанные на различных языках аспектов, а также экспериментируя со старыми и новые языки АОП. Более низкие уровни Reflex стабильны; как их архитектура, так и реализация были подтверждены в ходе целого ряда экспериментов, в частности с моделями параллельного программирования.

12. Zero—a blend of static typing and dynamic metaprogramming

В статье описан новый язык программирования Zero - экспериментальный статически типизированный, полностью объектно-ориентированный язык программирования с рефлексией. Рефлективные особенности охватывают самоанализ, а также структурные и поведенческие рефлексии, отражающие средства включают безопасные замены методов и классов и детальную модификацию методов. Это позволяет программам на Zero быстро адаптироваться к требованиям времени выполнения. Отражение поведения реализуется с помощью обработчиков (хуков), которые могут быть присоединены ко всем языковым конструкциям на основе замыканий. Zero обеспечивает эффективную систему статической типизации с расширениями времени выполнения. Методы являются значениями первого класса и представляются как объекты, когда такое представление требуется. Используя такое представление, Zero обеспечивает элегантное использование статически типизированных методов высшего порядка.

13. Metaprogramming Applied to Web Component Deployment

Разработана метапрограмма, которая интегрирует существующие компоненты в архитектуру веб-запросов. Хотя авторы решили

реализовать протокол simpleXML для удобства чтения, полное соответствие SOAP и WSDL легко достижимо с помощью технологии метапрограммирования. Очевидно, метапрограммирование хорошо работает в этой области. В совокупности размер программы составляет лишь 2500 LOC. Это показывает одно из преимуществ метапрограммирования. С точки зрения архитектуры авторы отмечают, что системы метапрограммирования имеют хороший уровень абстракции. Инкапсулируя около 82000 LOC за чистым интерфейсом, Recoder предоставляет полное структурное дерево с полным семантическим анализом и поддержкой общих преобразований. Системы метапрограммирования делают сложные технологии компиляторов легкодоступными. Как таковые, они являются полезным строительным блоком для более крупных систем.

14. Hierarchical representation of legal knowledge with metaprogramming in logic

В данной статье описано представление юридических знаний в управляемой металогике. Авторы утверждают, что юридические знания многослойны, и поэтому одноуровневому языку представления недостает необходимой выразительности. Классификация позволяет авторам определить многоуровневую модель юридических знаний и однозначное соответствие с уровнями метапрограммирования в логике. Пример выражен на многоуровневом языке программирования металогики, который обеспечивает соглашение об именах и использует отражение между уровнями. Авторы показали, что их представления достаточно для отражения некоторых важных аспектов юридической аргументации. В частности продемонстрировали, как юридическая герменевтика, то есть принципы толкования различных правовых источников, таких как положения, дела и т.д., могут быть включены в представление.

15. Улучшенный sandboxing для Groovy скриптов

Автор исследует вопрос использования Groovy в качестве платформы для скриптинга на JVM. Он рассказывает о различных механизмах

интеграции и показывает, что за это приходится платить безопасностью. Обозначены несколько концепций, таких как кастомайзеры компиляции, которые упрощают изоляцию среды выполнения скриптов. Существующие кастомайзеры, доступные в дистрибутиве Groovy, и доступные готовые проекты по sandboxing'у в общем случае не могут гарантировать безопасность запуска скриптов (конечно, это зависит от пользователей и от того, откуда пришел скрипт).

Продемонстрировано, как можно обойти эти ограничения с помощью расширений для проверки типов. Расширения для проверки типов настолько функциональны, что вы можете создать собственные сообщения об ошибке во время компиляции скриптов. Более того, сделав все это и закэшировав скрипты, можно получить поразительное повышение производительности при исполнении скриптов.

16. Введение в использование FreeMarker в Spring MVC

Автор кратко рассказывает о FreeMarker. Приводит пример работы в Spring MVC – подключение зависимостей, установка конфигураций и создания шаблона.

17. FreeMarker шаблоны

Рассматривается Apache FreeMarker — это механизм шаблонов: библиотека Java для генерации текстового вывода (HTML-страницы, xml, файлы конфигурации, исходный код и.т.д. На вход подается шаблон, например html в котором есть специальные выражения, подготавливаются данные соответствующие этим выражением, а Freemarker динамически вставляет эти данные и получается динамически заполненный документ. Так же автор приводит пример использования и создания шаблона.

18. Using Groovy in a Filter Expression

Приводится пример динамической вставки Groovy выражений в Security XML файл для получения доступа к объекту аутентификации.

19. Метапрограммирование

Метапрограммирование — это парадигма построения кода информационной системы с динамическим изменением поведения или структуры в зависимости от данных, действий пользователя или взаимодействия с другими системами. Задачи метапрограммирования: повышение абстракции кода и его гибкости, повторное использование, ускорение разработки, упрощение межсистемной интеграции. Техники метапрограммирования – компилируемые, интерпритируемые и гибридные.

20. Learning the Spring Expression Language

Статья посвящена основам работы со SpEL. Автор рассказывает о основных функциях и синтаксисе Spring Expression Language на нескольких небольших примерах. SpEL – это одна из мощных особенностей Spring framework. Язык выражений может быть применен к различным областям приложения, разработанного на Spring.

21. Анализ и систематизация материала.

Таблица 2. Результат анализа материала

№ источника	Научная статья	Год публикации	Динамическая интерпретация кода	Использование регулярных выражений	Имеет прямое отношение к МД	Присутствует исходный код
1	Да	2015	Да	Нет	Да	Нет
2	Нет	2018	Да	Да	Да	Да
3	Учебное пособие	2016	Нет	Нет	Нет	Нет
4	Да	2018	Да	Нет	Нет	Да
5	Нет	2012	Да	Нет	Да	Нет
6	Да	2016	Да	Нет	Нет	Нет
7	Да	2012	Да	Да	Нет	Нет
8	Нет	2014	Да	Нет	Да	Да
9	Да	2020	Нет	Да	Да	Нет
10	Да	2019	Да	Нет	Да	Нет
11	Да	2018	Да	Нет	Да	Нет
12	Да	2019	Да	Нет	Нет	Нет

13	Да	2012	Да	Да	Нет	Нет
14	Да	2004	Да	Нет	Нет	Нет
15	Нет	2019	Да	Нет	Да	Да
16	Нет	2020	Да	Нет	Да	Да
17	Нет	2018	Да	Нет	Да	Да
18	Нет	2019	Да	Да	Да	Да
19	Нет	2014	Да	Нет	Да	Да
20	Нет	2018	Да	Нет	Да	Да

Выводы

Наиболее полезными являются источники 2 и 18. Алгоритмы, которые используется в данных источниках, можно использовать как примеры для дальнейшей модификации в соответствии с целью и задачами магистерской диссертации.

Скорректированная цель и задачи

Цель: Разработать алгоритм фильтрации csv-файлов с возможностью динамической конфигурации правил и параметров фильтров.

Задачи:

1. Провести обзор существующих технологий динамической интерпретации кода
2. Спроектировать алгоритм фильтрации csv-файлов
3. Сформировать метрики производительности алгоритма
4. Реализовать прототипы с использованием технологий динамической интерпретации кода
5. Провести тестирование в условиях высоконагруженного трафика
6. Сравнить метрики производительности полученных прототипов алгоритма фильтрации

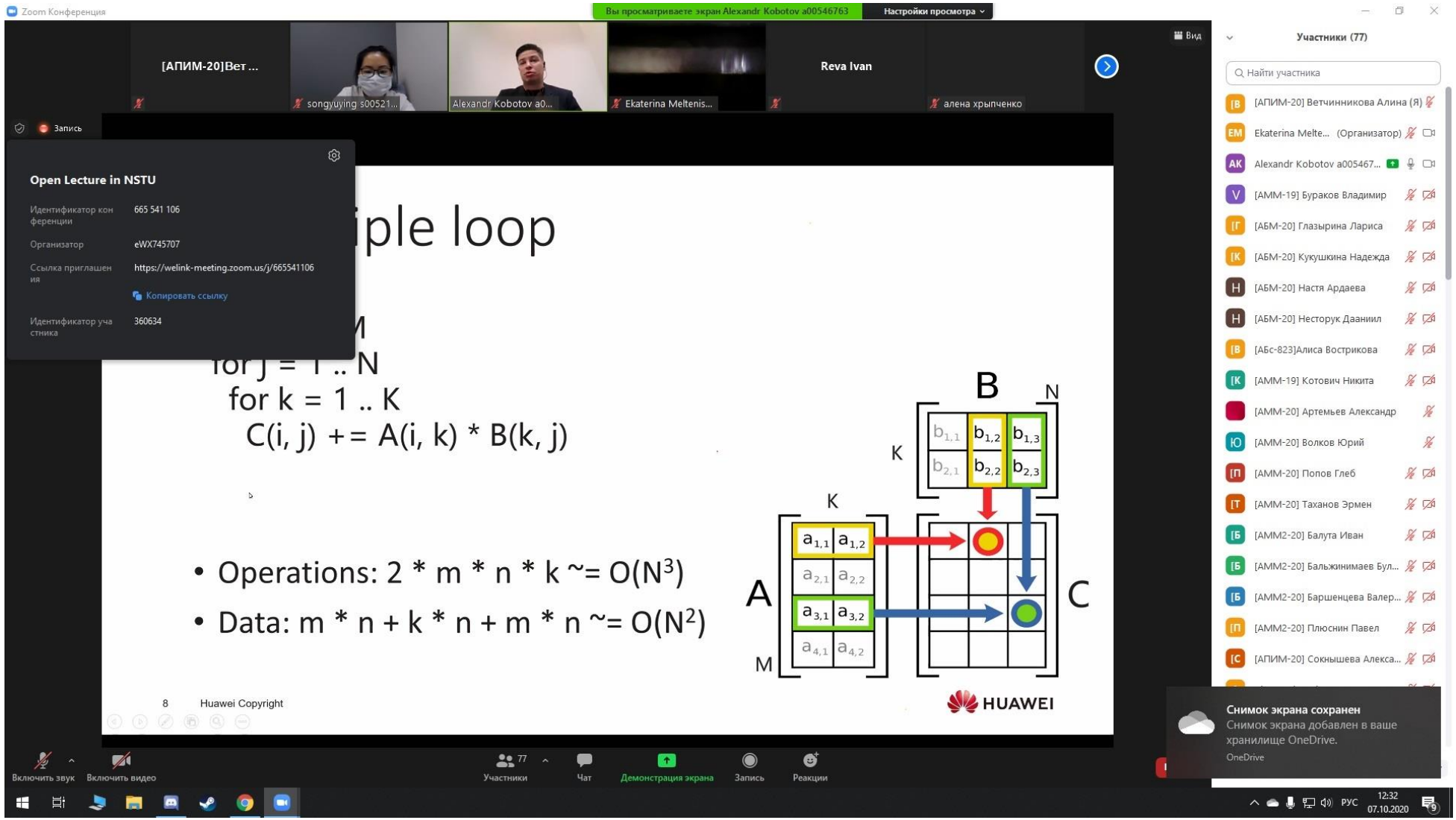
Список источников:

1. Benjamin Behringer, Eric Wagner, Towards Frame-based Source Code Generation for Heterogeneous Real-time Environments, IFAC Proceedings Volumes, Volume 45, Issue 4, 2015, p. 212-217
2. SpEL injection [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/154891/
3. Oracle Database Programming using Java and Web Services: /DonaldKnuth, 2016, p. 287-315
4. Benfano Soewito, Fergyanto E. Gunawan, Hirzi, Frumentius. Prevention Structured Query Language Injection Using Regular Expression and Escape String// Procedia Computer Science – 135, 2018, p. 678–687
5. Динамическая интерпретация метамоделей [Электронный ресурс]. - Режим доступа: https://habr.com/ru/company/dsec/blog/433034/
6. H.S. Yazdi, L.Angelis, T. Kehrer, U. Kelter, A framework for capturing, statistically modeling and analyzing the evolution of software models, Journal of Systems and Software, Volume 118, 2016, p. 176-207
7. Tom Dinkelaker, Michael Eichberg, Mira Mezin, Incremental concrete syntax for embedded languages with support for separate compilation, Science of Computer Programming, 78 (2013), p. 615-632
8. Метапрограммирование с примерами на JavaScript [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/227753/
9. Alexander Sakharov, Annotated regular expressions and input-driven languages – Information Processing Letters, Volumes 159–160, July 2020, 105958
10. Ignacio Lagartos, Jose Manuel Redondo, Francisco Ortin // Efficient runtime metaprogramming services for Java — Journal of Systems and Software, Volume 153, July 2019, Pages 220-237
11. Éric Tanter, Rodolfo Toledo, Guillaume Pothier, Jacques Noyé, Flexible metaprogramming and AOP in Java // Science of Computer Programming, Volume 72, Issues 1–2, 1 June 2018, p. 22-30

12. Sašo Greiner, Janez Brest, Viljem Žumer, Zero — a blend of static typing and dynamic metaprogramming – Computer Languages, Systems & Structures, Volume 35, Issue 3, October 2009, Pages 241-251
13. Welf Löwe, Markus L. Noga, Metaprogramming Applied to Web Component Deployment/ Electronic Notes in Theoretical Computer Science, Volume 65, Issue 4, April 2012, Pages 106-116
14. Jonas Barklund, Andreas Hamfelt, Hierarchical representation of legal knowledge with metaprogramming in logic – The Journal of Logic Programming, Volume 18, Issue 1, January 2004, Pages 55-80
15. Улучшенный sandboxing для Groovy скриптов [Электронный ресурс]. - Режим доступа: https://habr.com/ru/company/haulmont/blog/445114/
16. Введение в использование FreeMarker в Spring-MVC [Электронный ресурс]. - Режим доступа: https://www.codeflow.site/ru/article/freemarker-in-spring-mvc-tutorial
17. FreeMarker шаблоны [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/420549/
18. Using Groovy in a Filter Expression [Электронный ресурс]. - Режим доступа: https://community.jaspersoft.com/wiki/using-groovy-filter-expression
19. Метапрограммирование [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/137446/
20. Learning the Spring Expression Language (SpEL) [Электронный ресурс]. - Режим доступа: https://dzone.com/articles/learn-spring-expression-language-with-examples

Приложение

лекция:



Конференция