**Слайд 1.**

Здравствуйте! Меня зовут Инга. Я расскажу вам о своей работе по разработке средства визуального программирования для описания фрагментированных алгоритмов.

**Слайд 2.**

Для начала давайте разберемся что такое фрагментированное программирование и зачем оно нужно.

Фрагментированное программирование призвано автоматизировать решение задач системного параллельного программирования при реализации численных моделей. Идея этого подхода заключается в разделении параллельной программы на алгоритм и реализацию.

Алгоритм представляется в специальном виде – в виде двух множеств – множества фрагментов данных, множества фрагментов вычислений и отношений на этих двух множествах. Разделение на фрагменты сохраняется до самого момента исполнения. В таком представлении содержатся только зависимости по данным - это позволяет исполнительной системе динамически выполнять настройку на имеющиеся ресурсы.

Реализацию алгоритма обеспечивает специальная исполнительная система, которая автоматически исполняет алгоритм

**Слайд 3.**

Моей задачей является визуализация фрагментированного подхода к программированию. Сюда входит разработка визуального представления фрагментированного алгоритма, реализация визуальной среды разработки и транслятор для перевода графического представления в текстовое на языке LuNA.

**Слайд 4.**

Был выполнен обзор средств визуального программирования.

Языки визуального программирования делятся на диаграмматические и иконические.

Диаграмматические характеризуются строго определенным, формализованным словарем, состоящим из сравнительно небольшого количества элементов. Также, как правило, строго определен пространственный синтаксис – расположение элементов схем в пространстве и относительно друг друга.

Иконические языки используют иконы (пиктограммы) для представления объектов, операций и функций.

Так как достаточно сложно передавать всю информацию через изображения, иконические языки сильно уступают диаграмматическим. Как правило используются для обучения детей программированию.

Минусы всех систем:

* Громоздкое изображение — не помещается на экран
* Ограниченность языка

**Слайд 5.**

Проанализировав подходы к визуальному программированию мы можем определить что наш язык визуального программирования должен быть диаграмматическим. Иконический нам не подойдет.

(Если спросят: Разрабатываемый язык визуального программирования должен быть диаграмматическим. Это объясняется следующими различиями: в иконическом языке модель собирается из большого числа готовых элементов (иконок), а в диаграмматическом языке модель конструируется из небольшого числа базовых параметризованных элементов путем их соединения и задания конкретных значений параметров. Кроме того, диаграмматические языки позволяют иерархически создавать новые модели из уже существующих.)

Также данной системе не страшны основные проблемы визуальных языков, которые я уже перечислила. Это ограниченность языка, но для нас это не является проблемой, так как модель достаточно простая и мы можем ее представить в графическом виде без ограничений. И громоздкость схем, которая решаема сворачиванием описания.

**Слайд 6.**

Был разработан проект системы визуального программирования. Спроектирована среда разработки. Среда разработки позволяет… и т.д. по тексту слайда.

**Слайд 7.**

Система визуального программирования была реализована на базе платформы Eclipse при помощи двух фреймворков – Eclipse Modeling Framework и Graphical Editing Framework.

К особенностям относится то что сигнатуры внешних фрагментов вычислений считываются из отдельного файла и на их основе формируется правильно использование внешних ФВ в разрабатываемом алгоритме.

**Слайд 8.**

Здесь мы видим скриншот рассматриваемой системы визуального программирования. Справа в углу, желтая заметка – это комментарий, на котором представлен исходный фрагмент кода на языке LuNA. Этот фрагмент кода вычисляет y как сумму элементов массива x. На вход идет фрагмент данных x (вектор), так как на диаграмме в него не входит ничего, а на выход идет фрагмент данных y, так как на диаграмме из него ничего не выходит. Красными закругленными прямоугольниками обозначаются фрагменты данных, синими эллипсами – фрагменты вычислений. Сама диаграмма представляет из себя фрагмент кода. Справа находится палитра инструментов, слева в свернутом виде(чтобы изображение нормально вошло) располагается библиотека фрагментов кода. У связей между фрагментами данных и фрагментами вычислений есть 2 аннотации – индекс связанного ФД и имя аргумента связанного ФВ. Вот так выглядит сейчас данное средство разработки.

**Слайд 9.**

В среде визуального программирования Visual LuNA был реализован транслятор, выполняющий преобразование внутреннее представление алгоритма в представление, используемое системой фрагментированного программирования LuNA.

Транслятор был реализован в виде инкрементного компоновщика (incremental builder), для реализации которого платформа Eclipse предоставляет удобный интерфейс. При изменении файлов в проекте компоновщику поступает сигнал об изменении, содержащий информацию о том, какие файлы были изменены, и какие действия над ними были произведены (удаление, добавление, модификация). Используя эту информацию, компоновщик может осуществить сборку (трансляцию, компиляцию и т.п.) только тех файлов, которые подверглись изменению, не осуществляя лишний раз полную сборку проекта.

В нашей ситуации компоновщик транслирует, во-первых, EMF-модели, которые описывают диаграммы в .cfd файлах, и, во-вторых, объявления импортируемых фрагментов вычислений из файлов .imp, в байт-код языка LuNA. При возникновении ошибки на соответствующий файл устанавливается маркер с описанием ошибки и он не подвергается трансляции (сборка считается неуспешной).

Байт-код LuNA представлен в формате JSON (JavaScript Object Notation). По окончанию сборки компоновщик обновляет файл out.ja, находящийся в корневой папке проекта и содержащий байт-код LuNA. Далее этот файл можно передать исполнительной системе LuNA для исполнения.

В реализации транслятора использовалась свободная библиотека google-gson, позволяющая (де)сериализовывать объекты Java в JSON-представление и конструировать JSON-объекты произвольного вида.

**Слайд 10.**

На защиту выносятся...

Мной были проанализированы существующие подходы к визуальному программированию и на их основе и с учетом их недостатков было спроектировано визуальное представление фрагментированного алгоритма. При помощи Eclipse Modeling Framework и Graphical Editing Framework была реализована визуальная среда разработки фрагментированных алгоритмов. Был реализован транслятор из разработанной среды описания визуальных алгоритмов Visual LuNA в среду фрагментированного программмирования LuNA, а также была продемонстрирована возможность исполнения алгоритмов, описанных в среде Visual LuNA.

В ходе дальнейшей работы планируется расширить графический язык для

представления более сложных численных алгоритмов, а также интегрировать в среду Visual LuNA средства отладки и профилирования.

**Слайд 10.** На этом у меня все. Спасибо за внимание!

**Рецензия**

1. Люди, которые хоть немного знакомы с параллельным программированием, знают данные термины и понимают о чем идет речь.

Хорошая параллельная программа – программа которая хорошо реализует параллелизм заложенный в алгоритме.

Меньше памяти – больше памяти, есть видеокарта –нет видеокарты, разная топология сети - переносимость

1. Ошиблась
2. Не имеет принципиального значения как будут графически представляться компоненты алгоритма.
3. Если модель LuNA позволяет описывать реальные численные алгоритмы, то и моя модель позволяет это сделать.