# Вступление

Современная индустрия разработки ПО отличается от других наукоемких индустрий обилием и разнообразием используемых технологий. В наибольшей степени это касается языков программирования – даже если взять наиболее популярные решения в своей области, с легкостью можно найти около десятка аналогов. Поэтому, нередки ситуации, когда приложение или система разработаны с использованием нескольких языков программирования – в некоторых областях так сложилось исторически, в некоторых это необходимость так как нет других подходящих альтернатив.

Чаще всего взаимодействие между компонентами системы, реализованными на разных языках описано неявно или недостаточно явно. Это может быть связано с особенностями используемых технологий, кодогенерацией или с различной природой технологий, которые связываются вместе.

В любом случае, возникает необходимость поддержки таких систем с учетом разных используемых технологий, и проблема неявности связей мешает использовать стандартные средства для моноязыковых систем (дебаггеры, линтеры, плагины IDE) для выявления ошибок на этапе проектирования. Вследствие этого, баги и ошибки выявляются на этапе отладки или тестирования, что замедляет и усложняет процесс разработки. Одним из вариантов решения данной проблемы может стать использование статического анализа кода для выявления семантических связей между языками и использования такой информации для дальнейшей интеграции в IDE. В данной работе рассматривается метод такого мультиязыкового статического анализа.

# Цель работы

Цель данной работы – разработка универсального метода анализа, подразумевающего независимость относительно используемых языков программирования и гибкость в отношении быстродействия. Также, предложенный метод будет обеспечивать достаточную полноту анализа семантических связей для решения большого круга практических задач.

# Задачи

При разработке данного метода были решены следующие задачи:

* Рассмотрены различные существующие методы и подходы к статическому анализу
* Рассмотрен наиболее общий набор задач, решаемый большинством IDE при взаимодействии с мультиязыковыми проектами
* Проведен анализ особенностей связей мультиязыковых фрагментов кода (в особенности в современных веб-фреймворках)
* Выбран подход к универсальному статическому анализу и реализован минимальный прототип, использующий данный подход

# Результаты

В результате решения задач был синтезирован общий и относительно простой метод статического анализа, основанный на семантическом связывании компонентов. Основная идея метода – унификация информации о различных компонентах кода посредством единого представления. Каждый компонент (нередко в различных языках используется термин символ) описывается одним или несколькими узлами со следующей структурой:

* Позиция компонента в исходном коде (строка, столбец, файл);
* Атрибуты, выявленные при анализе компонента;
* Тип информации о компоненте;
* Семантическая роль компонента.

К примеру, компонент «функция на языке Си» может иметь следующую информацию:

* Позиция: 2 строка, 3 столбец, file.c
* Атрибуты: статическая, возвращает примитивное значение
* Тип: Си функция
* Семантическая роль: Объявление функции

Впоследствии, такая информация может быть связана с узлом, имеющим схожие атрибуты и семантическую роль «Использование функции». Таким образом, будет сформирована связь «Использование», которая позволит производить, например, навигацию по символу.

Предложенный метод был реализован в виде прототипа, который формировал граф зависимостей HTML тегов от JavaScript функций. Типами компонентов выступали “html-element” и “js-function”, а семантическими ролями были “Want” (использование) и “Give” (декларирование и определение). Такой онтологии было достаточно для связывания всех интерактивных html атрибутов и определений JavaScript функций. На слайде отражен пример связи между использованием функции getElementById и тегом from с id = regForm.

# Выводы и дальнейшее исследование

Предлагаемый метод обладает необходимой универсальностью и гибкостью для решения большего числа повседневных задач программистов. Метод не зависит от используемых технологий и может быть легко распараллелен при реализации, что может позволить получить быстродействие и отзывчивость, достаточную для интеграции метода в IDE.

У метода также имеется ряд особенностей:

* Необходимо формирование концептуальной схемы (онтологии) для проведения связывания компонентов.
* Метод предполагает получение семантической информации в первую очередь посредством парсинга изначальных фрагментов кода
* Сложность (как вычислительная, так и концептуальная) метода возрастает от сложности поставленной задачи и количества взаимосвязанных мультиязыковых компонентов

В качестве дальнейшего исследования необходимо рассмотреть возможности по формированию единой онтологии исходя из наиболее популярных парадигм программирования. Также, при анализе возможных вариантов использования такого метода имеет смысл проанализировать особенности современных мультиязыковых IDE, в особенности ориентированных на веб-разработку. В том числе, в связи со сложностью некоторых языковых конструкций, имеет смысл интегрировать методы машинного обучения для формирования атрибутов для семантических узлов.