УДК 004.4’232 004.4’413

© 2007 И.В. Русских

ИНКРЕМЕНТАЛЬНЫЙ СИНТАКСИЧЕСКИЙ АНАЛИЗ В СРЕДАХ РАЗРАБОТКИ И ТЕКСТОВЫХ РЕДАКТОРАХ

В статье рассматриваются решения по синтаксическому разбору структуры вводимого текста, применяемые в средах разработки для повышения эффективности работы программиста. Дается общее описание архитектуры библиотеки Colorer как альтернативного решения многих проблем во взаимодействии программиста со средой разработки.

Введение.

Разработка программного кода сегодня — огромная индустрия. Среды, упрощающие и поддерживающие процесс создания и отладки исходных кодов приложений, — один из наиболее востребованных на рынке видов программного обеспечения. Все лидирующие и популярные языки программирования распространяются в виде интегрированных сред, зачастую многофункциональных, предоставляющих и упрощающих программисту доступ к возможностям языка. Сложность современных архитектур, компиляторов, интерпретаторов требует от сред разработки все большей автоматизации, исключения из работы программиста рутинной технической работы.

Наличие интегрированной среды разработки существенно увеличивает эффективность и ускоряет процесс знакомства с языком [5, 2]. Вспомогательные средства: расцветка синтаксиса, выделение структуры исходного кода, контекстно-зависимые подсказки и помощь — уже давно неотъемлемые части любой крупной среды разработки.

К сожалению, разработка и внедрение поддержки новых языков программирования с использованием практически любой из существующих сред — сложная и трудоемкая задача. В данной работе представлена общая архитектура библиотеки Colorer — инструментария для реализации качественных и эффективных сред редактирования, поддерживающего большое количество существующих языков программирования, скриптов и разметок.

В отличие от обычных «мультиязычных» текстовых редакторов, расцветка текста программы — лишь базовая функциональность, предоставляемая библиотекой на основе синтаксического анализа текста программы. Процесс

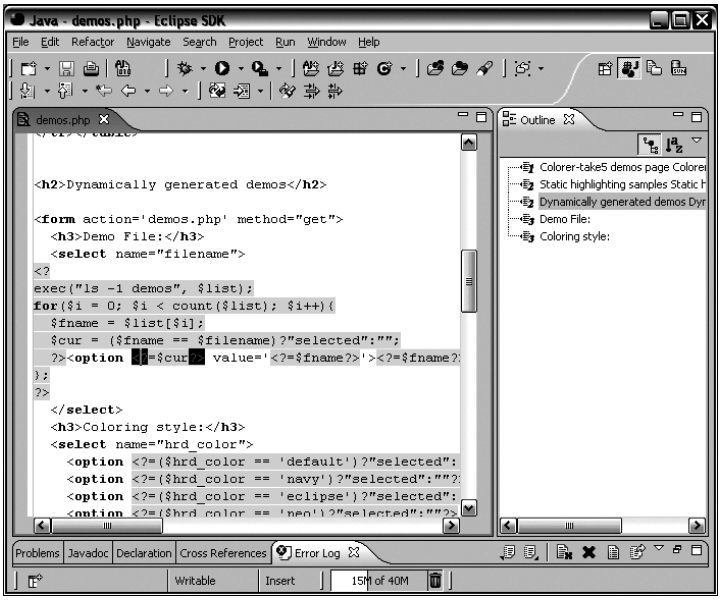

Рис. 1: EclipseColorer в среде Eclipse IDE

расцветки в библиотеке — это выделение значимых лексем и сопоставление им визуальных атрибутов, таких как цвет, шрифт, размер. Помимо визуальных атрибутов предоставляется анализ структуры текста: выделение структурного дерева, рекурсивных и парных конструкций позволяет целевой IDE реализовать расширенную навигацию по тексту (Рис. 1).

Библиотека Colorer не является полноценной пользовательской средой — это набор сервисов, «framework», который с легкостью интегрируется в целевые системы и среды разработки и предоставляет сервисы на основе анализа вводимого пользователем исходного текста.

Обзор существующих работ.

Наверное, одной из самых ранних систем, к которой можно применить термин «среда разработки», можно считать Cornell Program Synthesizer [6]. Эта система, по сути, представляет собой структурно-ориентированный редактор, позволяющий вводить текст программы (изначально на языках PL/CS и Pascal), используя шаблоны и структурное представление потока исполнения. Это позволяет оградить программиста от ошибок в синтаксисе и сконцентрироваться на логике программы.

Этот подход не особенно прижился, и в современных разработках в чистом виде структурно-ориентированные редакторы не встречаются. Развитие сред разработки пошло по пути текстового представления программ, которое лишь в последнее время стало расширяться различными визуальными вспо-

могательными средствами. Внимание стало уделяться каркасным средам и интегрированным решениям, библиотекам, позволяющим на основе базовых сервисов строить произвольные системы редактирования для различных целей и задач.

Популярная платформа Eclipse JDT [11] реализует свою систему редактирования на основе собственного инкрементального компилятора языка Java. Полноценный синтаксический и семантический разбор исходных модулей (Compilation Units) позволяет реализовать расцветку синтаксиса с учетом всех особенностей и найденных в тексте ошибок, автоматические шаблоны для структур языка, контекстно-зависимое автодополнение, рефакторинг кода, и т.д. Реализация большинства этих фундаментальных возможностей основывается на общем каркасном коде, Eclipse Framework, предоставляющем скелет и базовые средства для сред разработки целевых языков программирования. На основе этой же инфраструктуры сейчас развиваются такие проекты, как Eclipse CDT (для языков C/C++), Eclipse Web Tool (Среда для web-технологий) и многие другие.

Система Barista [1] делает упор на особенностях визуализации исходного кода, представляя его в виде древовидной объектной модели. Несмотря на структурно-ориентированное внутреннее представление исходного кода, эта система остается текстовой средой. Расширенные мультимедийные и интерактивные возможности вводятся в инструментарий программиста в виде вспомогательной функциональности, которая основывается на полном анализе текста в соответствии с описанием синтаксиса языка. Трудоемкость и полнота описания синтаксисов ограничивают эту систему пока лишь поддержкой языка Java.

Другое активно развивающееся исследование в отрасли систем редактирования, система Harmonia [8], избирает более общий подход. Harmonia генерирует базовую функциональность, скелет среды редактирования на основе общего описания контекстно-свободной грамматики целевого языка. Полученная отчасти сгенерированная, отчасти вручную написанная модель языка включает в себя лексический, синтаксический и семантический анализаторы исходного текста. Инкрементальный синтаксический анализ поддерживает соответствие между редактируемым текстом и генерируемым абстрактным синтаксическим деревом (AST). Это позволяет добиться развернутых возможностей, не выходя за рамки текстового редактора с интегрированным синтаксическим анализатором. Harmonia — одна из наиболее сложных систем в аспекте описания синтаксисов и добавления поддержки новых. Среда Harmonia поддерживает лишь несколько основных языков программирования, для которых в ней реализованы практически полноценные компиляторы и анализаторы семантической структуры.

Эти и другие направления разработки обычно уделяют внимание поддержке лишь некоторых базовых синтаксисов языков программирования. Се-

годня это «mainstream» языки — C, C++, C#, Java. Полноценная же реализация подобных сред для нового языка программирования (или группы языков) — трудоемкая и нетривиальная задача даже при наличии готовой инфраструктуры и скелета системы.

Отсутствие универсальных систем редактирования с поддержкой разнообразных языков программирования — большая проблема. Стоит отметить отсутствие мощных и доступных сред разработки для сред Web-программирования: PHP, Perl, Ruby, их шаблонных моделей для HTML и XHTML. Эти и подобные им технологии появляются, активно развиваются, но, к сожалению, разработка подходящих сред редактирования для них происходит с большой задержкой. При этом возникает необходимость в универсальных средах, позволяющих программисту в одном окружении работать со всем разнообразием документов его программного проекта.

Подобных систем существует большое количество, с разной степенью функциональности и качеством поддержки языков программирования. Одной из центральных возможностей при этом становится анализ структуры редактируемого исходного кода программы и расцветка его на лету (в процессе редактирования пользователем). Главный показатель здесь — универсальность. Не многие текстовые редакторы могут похвастаться большим списком поддерживаемых языков программирования. Во многих случаях количественные показатели никак не коррелируют с качеством реализации расцветки и анализа исходного кода. Часто разработчики многих систем просто не успевают поддерживать в приемлемом состоянии базы своих синтаксических описаний: слишком активно развиваются технологии, изменяются старые и появляются новые языки.

К мощным универсальным редакторам можно отнести такие системы, как Emacs, VIM, jEdit. Их описания синтаксисов и анализ структуры реализованы наиболее широко и полно. Встроенные в них синтаксические анализаторы на основе внутренних описаний грамматики языков выполняют работу по разбору текста и предоставляют на основе этой информации расширенные сервисы. Из универсальных библиотек, поддерживающих разнообразные языки и синтаксисы, следует отметить систему Scintilla [13].

Архитектура библиотеки Colorer.

Тенденции в развитии программных языковых средств сегодня таковы, что существующим средам разработки все тяжелее поддерживать их. Одна из таких тенденций — межъязыковая интеграция, происходящая на стыке различных технологий и стандартов. С развитием Web-технологий, сетевых, распределенных сред, активно стали появляться такие комбинированные синтаксисы, как ASP, JSP, PHP. Практически все популярные интерпретируемые языки обзавелись идентичными «шаблонными» моделями (PHP, Ruby on Rails). Синтаксическому анализатору необходимо на лету поддерживать

подобные «комбинированные» языки, состоящие из различных синтаксисов.

Аналогично развивается направление декларативных языков на основе XML. Этот стандарт, изначально предполагающий возможность смешения в одном документе различных концепций, привел к появлению действительно оригинальных языков и разметок (таких как XSLT) [17]. И хотя синтаксис XML тривиален, его поддержка ничего не дает для унаследованных от него языков и разметок. В них главную роль играет семантическая и структурная модель, которая накладывается поверх грамматической модели синтаксического анализатора. Концепция пространств имен XML позволяет при этом в одном XML документе использовать различные взаимосвязанные синтаксисы.

Основная проблема поддержки подобных языков в существующих средах — отсутствие возможности работать со смешанными синтаксисами. Хотя, теоретически, большинство таких синтаксисов определяются на основе правил контекстно-свободных грамматик, на практике использование существующих решений наталкивается на отсутствие гибких средств для работы с такими смешениями и описанием их в общепринятых BNF/EBNF.

В библиотеке Colorer основной акцент сделан на эффективном представлении и качественном анализе как подобных смешанных синтаксисов так и более традиционных языковых средств. Сегодняшние программные проекты обычно состоят из кода на базовых языках программирования, используют скриптовые языки для окружения проекта и его построения, конфигурационные скрипты, проектную документацию в различных форматах разметки. Под понятием *исходного текста*, языка, в библиотеке Colorer подразумеваются вообще любые подобные машинно-обрабатываемые синтаксисы.

Colorer предоставляет возможность интеграции и поддержки всего разнообразия языков в пределах одной программной системы. При этом разработчику предоставляется единый интерфейс редактирования с предсказуемым поведением. Это избавляет пользователя от изнурительного «подбора» подходящих, но разрозненных инструментов разработки для каждого из используемых в проекте языков программирования. Не являясь самостоятельным приложением, Colorer предоставляет набор объектно-ориентированных интерфейсов, использование которых позволяет интегрировать систему в любую среду редактирования и разработки.

Библиотеку выгодно отличает слабая связанность со средой на уровне программных интерфейсов: необходимо лишь реализовать интерфейс по предоставлению содержимого анализируемого документа, минимальную событийную модель для работы инкрементального синтаксического анализатора и обработчик дерева синтаксических структур, получаемых на выходе анализатора. В других существующих системах синтаксический анализатор зачастую тесно связан с внутренней моделью редактора, а это усложняет поддержку и развитие среды.

Древовидное представление исходного кода с выделенными токенами и контекстами сопоставляется затем с описанием стиля, каждому токену присваиваются визуальные и смысловые атрибуты: цвет, шрифт, стиль, тип токена. Полученная информация используется конечной средой редактирования для визуализации, построения структуры документа и т.д. Алгоритмы анализа, таким образом, не связаны с особенностями визуализации, что позволяет использовать библиотеку Colorer в широком спектре систем (консольных, графических).

Язык HRC. Основа функционирования синтаксического анализатора — сопоставление текста с набором синтаксических правил, описывающих грамматику целевого языка программирования. В библиотеке Colorer используется специально разработанный язык синтаксических описаний HRC[15].

Отличительной особенностью модели HRC и синтаксического анализатора Colorer является работа с анализируемым текстом не с целью жесткого его сопоставления заданной грамматике, а с целью вычленения регулярных и контекстно-свободных конструкций из общего потока символов в соответствии со специализированными грамматическими описаниями в HRC. Такой анализ отличается от классического тем, что он не останавливается на неизвестных и нераспознанных токенах, а продолжает работать дальше.

Его можно характеризовать как «позитивный» алгоритм, пытающийся выделить из текста структуру, в отличие от традиционных синтаксических анализаторов, выдающих ошибки в случае невозможности сопоставления текста заданному грамматическому описанию.

Язык HRC предоставляет расширение традиционных средств регулярных выражений и контекстов моделью наследования, виртуализации и параметризации. Основным инструментом описания правил разбора внутри одного модуля служат регулярные выражения для вычленения токенов текста и контексты — нетерминалы — для работы с контекстно-свободными рекурсивными конструкциями. На эту базовую модель накладывается концепция модульности, повторного использования. Контексты («scheme» в терминологии HRC) могут наследовать свойства других контекстов (определенные в них синтаксические правила), могут частично переопределять их поведение путем «виртуализации» контекстов, изменять их параметры. В совокупности эти возможности позволяют описывать многие сложные языковые конструкции, в том числе совсем не регулярные; при этом инкрементальный разбор ведется в реальном времени.

Всесторонняя поддержка смешанных синтаксисов, слабо поддающихся описанию в КС форме, возможна благодаря модуляризации описаний соответствующих синтаксисов, возможности динамического встраивания их друг в друга, переопределения и динамического изменения поведения существующих описаний синтаксисов. Язык HRC описывает каждый целевой синтаксис

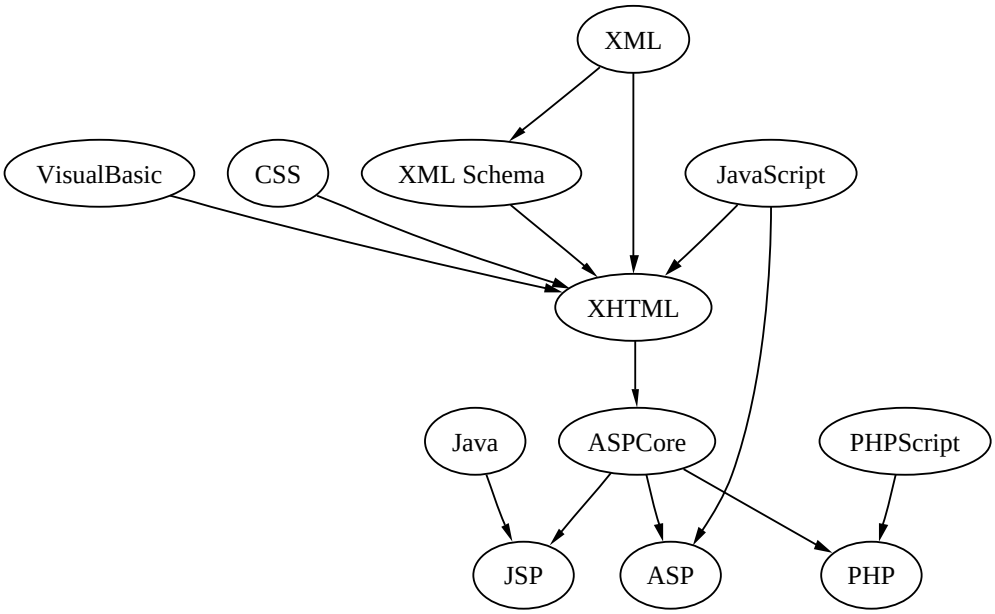

Рис. 2: Множество связанных языков в области Internet

как модуль, который затем может повторно использоваться в порожденных целевых синтаксисах. Таким образом определяются, например, раздельные описания для языков HTML, CSS, JavaScript, VBScript, ASP, которые затем комбинируются и используются взаимосвязано при разборе произвольного ASP файла, являющегося смешением этих синтаксисов (Рис. 2).

Помимо классических языков программирования (C, C++, Java, и т.д) сегодня следует отметить развитие языков с более сложными синтаксисами. Например, языки XML группы: XML Schema, XSLT/XSL, XPath, требуют от среды разработки поддержки семантики языка: контроль правильности (validation), корректной вложенности элементов, проверки атрибутов. Вручную описание таких моделей в любом контекстно-свободном представлении — сложная и зачастую бессмысленная работа.

За счет того, что язык HRC основан на синтаксисе XML, становится возможным расширение языковых средств для реализации конкретных задач в описании целевых синтаксисов. С использованием XSLT преобразований конструкции на специальных диалектах XML (XSD2HRC, Brackets) в библиотеке Colorer автоматически преобразуются в развернутые синтаксисы HRC, что с точки зрения разработчика повышает качество поддержки разрабатываемых синтаксисов, простоту их отладки и сопровождения.

Наиболее развитый компонент библиотеки, работающий в таком русле — XSLT модуль XSD2HRC, позволяющий в автоматическом и управляемом

ручном режимах получать готовые HRC описания для любых диалектов языка XML на основании информации о структуре: DTD или XML Schema. В библиотеке Colorer на основе этого модуля реализованы мощнейшие синтаксисы для языков XHTML, SVG, XML Schema, XPath, XSLT 1/2, RelaxNG, ANT, Docbook, MathML и многих других. Все они поддерживают проверку синтаксиса и семантической структуры документа, атрибутов и их значений, реализуют структурную модель для навигации по значимым тэгам и другие возможности.

Синтаксический анализатор. Инкрементальный синтаксический анализатор библиотеки Colorer не строит дерево разбора в традиционном его понимании. Во внутренней модели хранится лишь частичный кэш разбора, основывающийся на построчном разбиении исходного текста. Кэш хранит дерево разбора, игнорируя смены контекстов в пределах строки текста. Это позволяет оптимизировать использование памяти в процессе инкрементального анализа.

Анализатор так же не хранит раздельные токены внутри этого дерева — в нем используется «Push» модель, в которой внешние модули лишь получают нотификации о потоке токенов и смене контекстов через шаблон «Visitor» (интерфейсный абстрактный класс, поглощающий информацию от синтаксического анализатора).

Такая модель имеет ряд преимуществ по сравнению с традиционной моделью построения и поддержания дерева разбора непосредственно самим анализатором. Во-первых, клиент, получающий информацию о потоке разбора (текстовый редактор), может организовать удобную ему форму хранения визуальной и структурной информации о тексте. Дополнительные компоненты могут играть, например, роль фильтров, обрабатывая лишь нужные события из потока анализатора. Библиотека предоставляет базовую реализацию, хранящую полное дерево с отображением его в визуальные стили.

Еще одно преимущество такой модели — внутренняя логика работы инкрементального анализатора сильно упрощается, приближение дерева анализа до уровня строк документа позволяет эффективно им управлять и обновлять в соответствии с внешними событиями. Алгоритму синтаксического анализа не требуется поддерживать в актуальном состоянии все токены в месте модификации текста — дерево анализа основывается лишь на иерархии контекстов.

Анализатор работает по обобщенному top-down алгоритму, при этом его особенностью является работа непосредственно с моделью представления синтаксиса в HRC. Для этого в процессе инкрементального анализа строятся две иерархии: дерева обхода синтаксических описаний HRC и дерево структуры исходного текста. Эти деревья динамически связаны друг с другом и обновляются анализатором инкрементально при модификации текста

пользователем. Дерево синтаксических описаний используется при этом для реализации наследования и переопределения контекстов.

Инкрементальный анализ основывается на обобщенных событиях, сигнализирующих об изменении текста. Алгоритм пытается восстановить структуру анализа при точечной модификации (в пределах одной строки текста), при невозможности это сделать дерево перестраивается с места модификации и до конца текста.

Обзор функциональности.

На основании описанной выше архитектуры библиотеки Colorer и модели языка HRC реализуется широкий набор функциональности, предоставляемый программисту средой разработки.

Основное свойство — визуальная разметка и расцветка исходного кода программы. Для различных систем и сред разработано большое количество визуальных стилей (HRD), представление которых варьируется, начиная от цветовой гаммы и заканчивая начертанием и размером шрифта. Пользователю не составляет большого труда расширить набор этих стилей своими собственными.

На основе расширенного анализа и фильтрации информации от синтаксического анализатора, такие среды как Eclipse строят отдельную структуру для навигации по редактируемому документу (Outline). Эта функциональность поддерживается в описаниях практически всех основных языков в библиотеке Colorer.

Основываясь на инкрементальном построении дерева анализируемого документа, узловым элементам в тексте ставятся в соответствие регионы HRC (смысловые атрибуты). Пользователю на их основе предоставляется возможность навигации по структуре исходного кода, произвольным парным синтаксическим конструкциям.

Расширенный синтаксический анализ в некоторых из описаний синтаксисов (XML в частности) включает проверку семантики документа и представление несоответствий отдельным визуальным стилем. Помимо этого создаются навигационные списки таких некорректных элементов в документе, что позволяет программисту видеть основные синтаксические ошибки и быстро переходить к ним.

Перечисленной функциональностью пользуются конечные подключаемые модули Colorer, гибкость и переносимость библиотеки позволили внедрить библиотеку в среды Eclipse IDE, FAR Manager, Midnight Commander.

Заключение.

Благодаря полноценной поддержке более 150 языков и скриптов программирования, библиотека Colorer популярна как среди пользователей специализированных языков программирования (у которых отсутствует полнофункциональная IDE), так и среди пользователей «mainstream» технологий.

По некоторым данным (статистика сервера sourceforge.net) число активных пользователей версии библиотеки в среде Eclipse превышает 25000, в среде Far Manager — около 13000, в среде Midnight Commander — около 1000 человек.

Альтернативной областью использования библиотеки Colorer является работа в Web-приложениях, интеграция с online и blogging системами на основе Perl [14] и PHP. Система включена в официальный набор ПО некоторых дистрибутивов ОС Linux.

Список литературы

[1] Andrew J. Ko and Brad A. Myers. Barista: An Implementation Framework for Enabling New Tools, Interaction Techniques and Views in Code Editors. CHI 2006 Proceedings, 387-396.

[2] Thomas, Richard C. Long Term Human-Computer Interaction, Springer-Verlag, 1998.

[3] Русских И.В. Оптимизация системы динамического распределения памяти в приложениях на примере библиотеки Colorer-take5. // Вестник Нижегородского университета, сер. Математическое моделирование и оптимальное управление, Н.Новгород, вып. 1(27), 2004. с.234-242.

[4] Русских И.В. Основные принципы разработки библиотеки Colorer. // Методы и средства обработки сложной графической информации. Тезисы докладов. Н.Новгород, 2003 г. с. 80-82.

[5] Roberts, Teresa L. and Moran, Thomas P. The Evaluation of Text Editors: Methodology and Empirical Results. Communications of the ACM , April, 1983.

[6] Teitelbaum, T. and Reps, T., The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, CACM, 24, 9, (1981), 563-573.

[7] Rhodes, C., Strandh, R., Mastenbrook, B. Syntax Analysis in the Climacs Text Editor, 2005.

[8] Boshernitsan, M., Harmonia: A Flexible Framework for Constructing Interactive Language-Based Programming Tools, University of California, Berkeley, Technical Report CSD-01-1149, 2001.

[9] Steven R. Wood, Z — the 95% program editor, ACM SIGPLAN Notices, v.16 n.6, p.1-7, June 1981

[10] Столяров А. В. Интеграция разнородных языковых механизмов в рамках одного языка программирования : Дис. канд. физ.-мат. наук: 05.13.11 М.: 2002, 105 с.

[11] Eclipse IDE. http://www.eclipse.org.

[12] Stallman, R. M. EMACS, The Extensible, Customizable, Self-Documenting Display Editor. Technical Report AI Memo 519, Massachusetts Institute of Technology, June, 1979.

[13] Neil Hodgson and contributors, Scintilla. http://www.scintilla.org.

[14] Perl Syntax-Highlighting-Universal.

http://search.cpan.org/∼palant/Syntax-Highlight-Universal.

[15] Igor Russkih. HRC Language Reference, 2004. http://colorer.sf.net/hrc-ref/

[16] Extensible Markup Language (XML) 1.0 Second Edition. Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen, Eve Maler, editors. W3C Recommendation, 2000. (http://www.w3.org/TR/2000/REC-xml-20001006)

[17] XSL Transformations (XSLT). James Clark, editor. W3C Recommendation, 1999. http://www.w3.org/TR/xslt

Нижегородский государственный университет им. Н.И.Лобачевского
603950, Нижний Новгород, пр. Гагарина, 23

INCREMENTAL SYNTAX ANALYSIS IN THE INTEGRATED DEVELOPMENT ENVIRONMENTS AND PROGRAMMING EDITORS

© 2006 Igor Russkih
Nizhny Novgorod State University
23 Gagarin Ave., 603950 Nizhny Novgorod, Russia

The article reviews a number of existing solutions for real-time syntax analysis in the Integrated Software Development Environments (IDE) targeting software engeneer's effeciency improvements. Colorer library is presented as an alternative solution in this area for many of the interaction problems between the developer and development environment.