srcML & srcDiff:

Infrastructures to Support the Software Evolution Community

Michael J. Decker
mdecke@bgsu.edu
Department of Computer Science
Bowling Green State University
Ohio, USA

BGSU

SES 2024 国際講演

srcML (sɔɹs ɛm ɛl), *n*.

1 : an infrastructure for the exploration, analysis, and manipulation of source code.

2 : an XML format for source code.

3 : a lightweight, highly scalable, robust, multi-language parsing tool to convert source code into srcML.

4 : an open source software application licensed under GPL.

srcML Infrastructure

Tools

- Tools provided and custom built are used to query, extract data, and transform source code.

Models

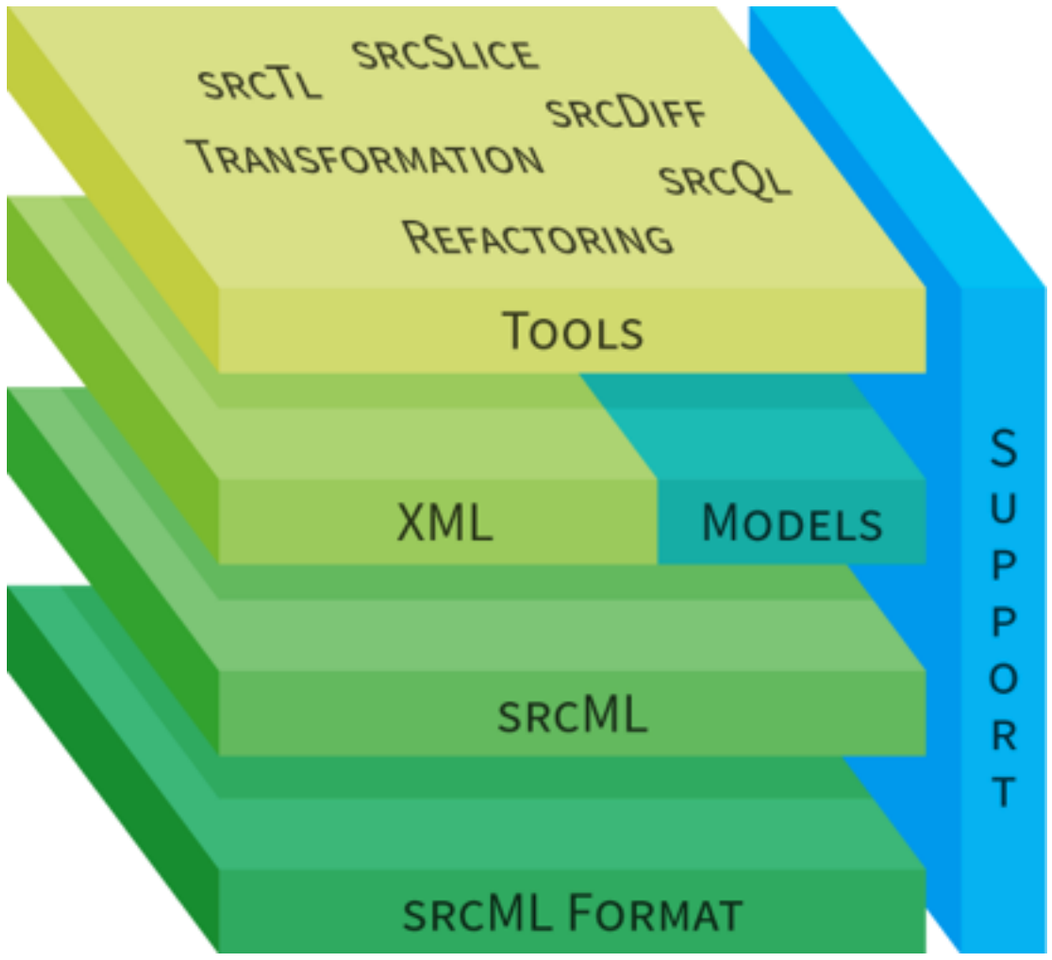
- External models of the code such as PDG, UML, call graphs can be built in XML

XML

- The full range of XML technologies can be applied to the srcML format.

srcML

- The srcml CLI is used to convert entire projects from and to source code and the srcML format. Languages supported include C, C++, Java, and C#.



srcML Format

- The srcML format represents source code with all original information intact, including whitespace, comments, and preprocessing statements.

Support

- A multi-university team currently supports the infrastructure.

The srcML Format

- A document-oriented XML format that explicitly embeds structural information directly into the source text
- Markup is selective at a high Abstract Syntax Tree (AST) level
  - no sub-expressions
- All original text preserved, including white space, comments, special characters

Source Code

```
#include "rotate.h"

// rotate three values
void rotate(
                int& n1,
                int& n2,
                int& n3
) {
    // copy original values
    int tn1 = n1,
        tn2 = n2,
        tn3 = n3;

    // move
    n1 = tn3;
    n2 = tn1;
    n3 = tn2;
}
```

srcML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<unit xmlns="http://www.srcML.org/srcML/src" xmlns:cpp="http://www.srcML.org/srcML/cpp" revision="1.0.0" language="C++">

<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>"rotate.h"</cpp:file></cpp:include>

<comment type="line">// rotate three values</comment>
<function><type><name>void</name></type> <name>rotate</name><parameter_list>(
            <parameter><decl><type><name>int</name><modifier>&amp;</modifier></type> <name>n1</name></decl></parameter>,
            <parameter><decl><type><name>int</name><modifier>&amp;</modifier></type> <name>n2</name></decl></parameter>,
            <parameter><decl><type><name>int</name><modifier>&amp;</modifier></type> <name>n3</name></decl></parameter>
)</parameter_list>
<block>{<block_content>
    <comment type="line">// copy original values</comment>
    <decl_stmt><decl><type><name>int</name></type> <name>tn1</name> <init>= <expr><name>n1</name></expr></init></decl>,
        <decl><type ref="prev"/><name>tn2</name> <init>= <expr><name>n2</name></expr></init></decl>,
        <decl><type ref="prev"/><name>tn3</name> <init>= <expr><name>n3</name></expr></init></decl>;</decl_stmt>

    <comment type="line">// move</comment>
    <expr_stmt><expr><name>n1</name> <operator>=</operator> <name>tn3</name></expr>;</expr_stmt>
    <expr_stmt><expr><name>n2</name> <operator>=</operator> <name>tn1</name></expr>;</expr_stmt>
    <expr_stmt><expr><name>n3</name> <operator>=</operator> <name>tn2</name></expr>;</expr_stmt>
</block_content>}</block></function>
</unit>
```

What does srcML do?

- Convert source code to srcML
- Convert srcML back to original source, with ***no*** loss of text
- Query code using XML query languages, such as XPath
- Transform source code while in srcML format

src ➡ srcML ➡ query/transform ➡ srcML ➡ src

Implementation

- Parsing technology in C++ with ANTLR
- Uses *libxml2, libarchive*
- Current speed: 262 KLOC/second
- Linux Kernel: 2 minutes
- srcML to text: ~3.4 (~1.5 compressed)
- Allows for various input sources
  - Directories, source archives (*tar.gz,* etc)

Language Support

- C11, K&R C
- C++17, Qt extensions
- Java SE 8
- C# Standard ECMA-334
- OpenMP pragmas

Applications of srcML

- Static analysis: slicing, pointer analysis, PDG, etc.
- Fact extraction, custom profiling
- Computing metrics
- Refactoring, transformation
- Syntactic differencing
- Reverse engineering UML class diagrams, method/class stereotypes, and C++ template parameter constraints
- C++ preprocessor analysis

Using srcML

SES 2024

Using srcML

- foo.cpp ➡ ***srcml*** + XPath

- foo.cpp ➡ ***srcml*** ➡ foo.cpp.xml ➡
  - ***XML Tools*** (e.g., XSLT, XPath)
  - application code + ***libxml2***
  - ***srcSAX*** framework

- foo.cpp ➡ application code + ***libsrcml*** ➡
  - ***XML Tools*** (e.g., XSLT, XPath)
  - application code + ***libxml2***
  - ***srcSAX*** framework

Query srcML with XPath

- Names of all functions that include a direct call to *malloc()*:

```
% srcml --xpath="//src:function[.//src:call/
  src:name='malloc']/src:name"
    linux-4.0.3.tar.xz.xml -o function_names.xml
```

- Result: srcML Archive with `<unit>` for each function name

- Good for collecting results in isolation

Using XSLT

- Can also use other XML technologies

- Custom instrumentation tool for profiling
- Search through the srcML, insert instrumentation code as needed

- Count the number of times each function is called

```
<!-- Insert includes and declarations for file with main() function -->
<xsl:template match="src:unit[src:function[./src:name='main']]">
                                    <xsl:copy><xsl:apply-templates select="@*"/>
<xsl:text>#include "profile.hpp"
profile functions;
</xsl:text><xsl:apply-templates select="node()"/></xsl:copy></xsl:template>

<!-- Insert includes and declarations for file without main() function -->
<xsl:template match="src:unit[not(src:function[./src:name='main'])]">
                                            <xsl:copy><xsl:apply-templates select="@*"/>
<xsl:text>#include "profile.hpp"
extern profile functions;
</xsl:text><xsl:apply-templates select="node()"/></xsl:copy></xsl:template>

<!-- Insert report output at last return of main() -->
<xsl:template match="src:return[ancestor::src:function[./src:name='main'] and last()]">
<xsl:text>std::cout &lt;&lt; functions &lt;&lt; std::endl;
</xsl:text><xsl:copy-of select="."/></xsl:template>

<!-- Insert call to global profile object to record call of this function -->
<xsl:template match="src:function/src:block">
<xsl:copy-of select="node()[1]"/>
<xsl:text>functions.count(__LINE__, "</xsl:text><xsl:value-of select="../src:name"/><xsl:text>");
</xsl:text><xsl:apply-templates select="node()[position()!=1]"/></xsl:template>

<!-- identity copy -->
<xsl:template match="@*|node()">
<xsl:copy><xsl:apply-templates select="@*|node()"/></xsl:copy>
</xsl:template>
```

Using libxml2

- Can use libxml2 to develop custom applications
- Python, Java, C++, etc.

- Print a list of all function calls made within each function
- Give max and average


```
reader = libxml2.newTextReaderFilename(sys.argv[1])
max_calls = 0
total = 0
count = 0
reader.Read()
while reader.Read():
  if reader.NodeType() == 1 and reader.Name() == 'function':
    # expand the subtree
    node = reader.Expand()

    # setup for XPath evaluation
    ctxt.setContextNode(node)

    # output the function name
    print ctxt.xpathEval("src:name")[0].getContent() + ',',

    # output the calls
    calls = ctxt.xpathEval("src:block//src:call/src:name")
    calllist = [call.getContent() for call in calls]

    count = count + 1
    max_calls = max(max_calls, len(calllist))
    total = total + len(calllist)

    # output the list of calls
    print ','.join(set(calllist))

    # delete this subtree
    reader.Next()

ctxt.xpathFreeContext()
print "Max: " + str(max_calls)
print "Avg: " + str(total / count)
```

libsrcml

- C-based API for srcML translation, querying, and transformation
- *srcml* – CLI client
- *libsrcml* – C API

- *libsrcml* allows you to build srcML features into your code

- Wrappers available for Python & JavaScript

Convenience API

```
#include <srcml.h>

int main() {

  // main.cpp -> main.cpp.xml
  srcml("main.cpp", "main.cpp.xml");

  // main.cpp.xml -> newmain.cpp
  srcml("main.cpp.xml", "newmain.cpp");

  return 0;
}
```

src to srcML

```
// setup the overall srcML output
srcml_archive* srcml_arch = srcml_archive_create();
srcml_archive_write_open_filename(srcml_arch, "main.cpp.xml");

// create and write the unit
srcml_unit* unit = srcml_unit_create(srcml_arch);
srcml_unit_set_filename(unit, "main.cpp");

srcml_unit_parse_filename(unit, "main.cpp");

srcml_archive_write_unit(srcml_arch, unit);
srcml_unit_free(unit);

// cleanup
srcml_archive_close(srcml_arch);
srcml_archive_free(srcml_arch);
```

srcML to src

```
// setup the overall srcML input
srcml_archive* srcml_arch = srcml_archive_create();
srcml_archive_read_open_filename(srcml_arch, "main.cpp.xml");

// read the unit and write to filesystem
srcml_unit* unit = srcml_archive_read_unit(srcml_arch);
srcml_unit_unparse_filename(unit, "newmain.cpp");

srcml_unit_free(unit);

// cleanup
srcml_archive_close(srcml_arch);
srcml_archive_free(srcml_arch);
```

srcSAX

- Event driven framework for efficient processing of srcML
- Used internally in *libsrcml*

- *C Version*
- C++ Handler

srcSAX Overrides

```
class srcSAXHandler {
public :
    virtual void startDocument() {}
    virtual void endDocument() {}
    virtual void startRoot() {}
    virtual void endRoot() {}
    virtual void startUnit() {}
    virtual void endUnit() {}
    virtual void endFunction() {}
    virtual void startFunction() {}
    virtual void startElement() {}
    virtual void endElement() {}

    virtual void charactersRoot() {}
    virtual void charactersUnit() {}
    virtual void metaTag() {}
    virtual void comment() {}
    virtual void cdataBlock() {}
    virtual void processingInstruction() {}
};
```

Example: Count Elements

```
class element_count_handler : public srcSAXHandler {

private :

    /** map to count srcML elements */
    std::map<std::string, long> element_counts;

public :

    /**
     * get_counts
     *
     * Accessor method to get the element counts.
     *
     * @returns the element count map.
     */
    const std::map<std::string, long> & get_counts() const {

        return element_counts;

    }
```

update_count

```
void update_count(const char * prefix, const char * localname) {

        std::string element;
        if(prefix) {
            element += prefix;
            element += ":";
        }
        element += localname;

        std::map<std::string, long>::iterator
                                    itr = element_counts.find(element);
        if(itr == element_counts.end()) {
            element_counts.insert(std::pair<std::string, long>(element, 1));
        } else {
            ++itr->second;
        }

}
```

Overloads

```
virtual void startRoot(const char * localname, const char * prefix, const char * URI,
                       int num_namespaces, const struct srcsax_namespace * namespaces,
                       int num_attributes, const struct srcsax_attribute * attributes) {
        if(is_archive)
            update_count(prefix, localname);
}
```

```
virtual void startUnit(const char * localname, const char * prefix, const char * URI,
                       int num_namespaces, const struct srcsax_namespace * namespaces,
                       int num_attributes, const struct srcsax_attribute * attributes) {
            update_count(prefix, localname);
}
```

```
virtual void startElement(const char * localname, const char * prefix, const char * URI,
                          int num_namespaces, const struct srcsax_namespace * namespaces,
                          int num_attributes, const struct srcsax_attribute * attributes) {
        update_count(prefix, localname);
}
```

main

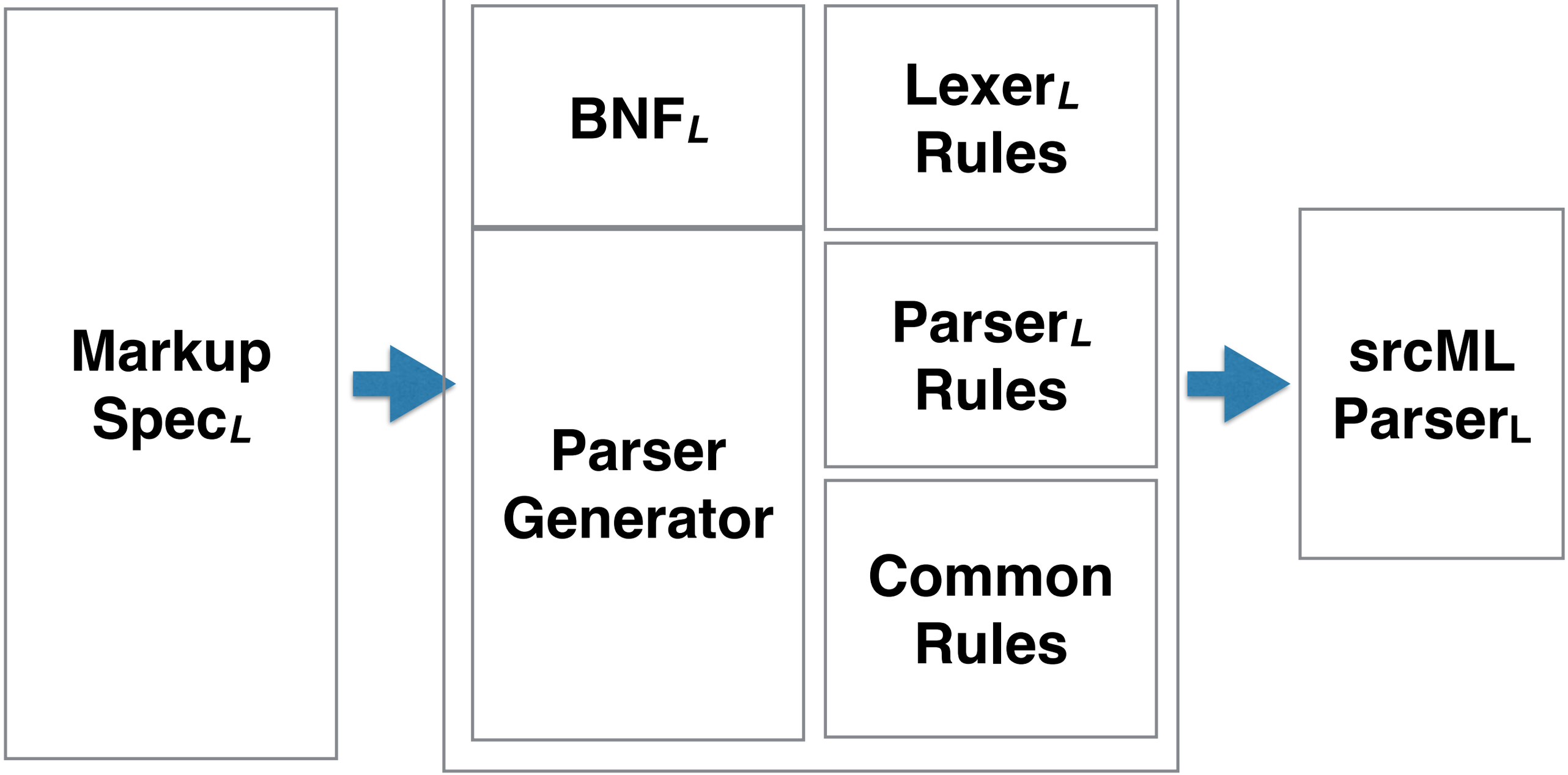
```
#include "element_count_handler.hpp"
#include <srcSAXController.hpp>

int main(int argc, char * argv[]) {

  srcSAXController control(argv[1]);
  element_count_handler handler;
  control.parse(&handler);

  for(map<string,long>::iterator citr = handler.get_counts().begin();
      citr != handler.get_counts().end();
      ++citr) {

       std::cout << citr->first << ": " << citr->second << '\n';

  }
  return 0;
}
```

Supporting New Programming Languages

New Language Support

- Moving forward want to support more languages
- How do get this accomplished:
  - Parser generator
  - For each language
    - Markup specification
    - Parser test cases

Parser Generator



Markup Specification

- Write expected srcML markup for each structural element including variations

- Write Permissive grammar/specification - accepts a superset of the language

- Always parsing correct code so no syntax checking is necessary

Examples

```
<for> for <control> <name/> <range>
            in <expr/> </range> </control> <block/>
</for>
```

```
<for> for <control> <name/> <range>
            in <expr/> </range> </control> <block/>
            <else> else <block/> </else>
</for>
```

```
<for> <specifier>async</specifier>
      for <control> <name/> <range>
            in <expr/> </range> </control> <block/>
</for>
```

New Languages

- Python - specification and test cases finished
- Swift - 99% of specification done
- Javascript - mostly done

- Rust
- Go
- Ruby
- ?

Tools Built with srcML

SES 2024

Tools (beta release)

- Playground WASM
- srcSlice - highly scalable forward static slicer
- srcPtr - lightweight pointer analysis tool
- srcType - static type resolution
- srcUML - Source to UML class diagrams
- stereoCode - method/class stereotypes
- nameCollector – identifier names

Tools (in development)

- srcQL - syntactic aware query language
- srcTL - transformation language
- srcNLP - identifier part of speech tagger

WASM

Playground

src ○ C ◉ C++ ○ C# ○ Java

Choose File srcml.cpp

srcML

Hash Attribute Download

```
1  // SPDX-License-Identifier: GPL-3.0-only
2  /**
3   * @file srcml.cpp
4   *
5   * @copyright Copyright (C) 2014-2019 srcML, LLC. (www
6   *
7   * This file is part of the srcml command-line client.
8   */
9
10 #include <srcml.h>
11 #include <srcml_cli.hpp>
12 #include <srcml_options.hpp>
13 #include <create_srcml.hpp>
14 #include <compress_srcml.hpp>
15 #include <create_src.hpp>
16 #include <srcml_display_metadata.hpp>
17 #include <srcml_execute.hpp>
18 #include <Timer.hpp>
19 #include <SRCMLStatus.hpp>
20 #include <curl/curl.h>
21 #include <iostream>
22 #include <csignal>
23 #include <cmath>
24 #include <TraceLog.hpp>
25 #include <string_view>
26
27 #ifndef _MSC_VER
28 #include <sys/uio.h>
29 #include <unistd.h>
30 #endif
31
32 // Number representing srcML markup version
33 [[maybe_unused]] static const int SRCML_CLIENT_VERSION
34
35 // String containing srcML markup version
36 static const std::string_view SRCML_CLIENT_VERSION_STR
```

```
<unit xmlns="http://www.srcML.org/srcML/src" xmlns:cpp="http://www.srcML.org/srcML/cpp" revision="1.0.0" lang
<comment type="block" format="doxygen">/**
 * @file srcml.cpp
 *
 * @copyright Copyright (C) 2014-2019 srcML, LLC. (www.srcML.org)
 *
 * This file is part of the srcml command-line client.
 */</comment>

<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;srcml.h&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;srcml_cli.hpp&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;srcml_options.hpp&gt;</cpp:file></cpp:incl
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;create_srcml.hpp&gt;</cpp:file></cpp:inclu
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;compress_srcml.hpp&gt;</cpp:file></cpp:inc
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;create_src.hpp&gt;</cpp:file></cpp:include
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;srcml_display_metadata.hpp&gt;</cpp:file><
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;srcml_execute.hpp&gt;</cpp:file></cpp:incl
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;Timer.hpp&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;SRCMLStatus.hpp&gt;</cpp:file></cpp:includ
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;curl/curl.h&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;iostream&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;csignal&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;cmath&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;TraceLog.hpp&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;string_view&gt;</cpp:file></cpp:include>

<cpp:ifndef>#<cpp:directive>ifndef</cpp:directive> <name>_MSC_VER</name></cpp:ifndef>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;sys/uio.h&gt;</cpp:file></cpp:include>
<cpp:include>#<cpp:directive>include</cpp:directive> <cpp:file>&lt;unistd.h&gt;</cpp:file></cpp:include>
<cpp:endif>#<cpp:directive>endif</cpp:directive></cpp:endif>

<comment type="line">// Number representing srcML markup version</comment>
<decl_stmt><decl><attribute>[[<expr><name>maybe_unused</name></expr>]]</attribute> <type><specifier>static</s

<comment type="line">// String containing srcML markup version</comment>
<decl_stmt><decl><type><specifier>static</specifier> <specifier>const</specifier> <name><name>std</name><oper
```

srcDiff Infrastructure

SES 2024

srcDiff [Decker ’19]

- Syntactic differencing approach
- Does not use tree-edit distance
- Set of domain rules to compute difference
- Better mapping of programmer’s view of change

Example

Original Code

```
public static boolean isStoragePolicyXAttr(XAttr xattr) {
  return xattr != null && xattr.getNameSpace() == XAttrNS
      && xattr.getName().equals(STORAGE_POLICY_XATTR_NAME);
}
```

Modified Code

```
public static String getStoragePolicyXAttrPrefixedName() {
  return XAttrHelper.getPrefixedName(XAttrNS, STORAGE_POLICY_XATTR_NAME);
}
```

Shortest Tree-Edit Distance

Original Code

```
public static boolean isStoragePolicyXAttr(XAttr xattr) {
  return xattr != null && xattr.getNameSpace() == XAttrNS
      && xattr.getName().equals(STORAGE_POLICY_XATTR_NAME);
}
```

Modified Code

```
public static String getStoragePolicyXAttrPrefixedName() {
  return XAttrHelper.getPrefixedName(XAttrNS, STORAGE_POLICY_XATTR_NAME);
}
```

srcDiff Format [Decker ’19]

- srcML format with tags added to markup delta
- Lossless: comments, whitespace, and preprocessor preserved
- Supports srcML Infrastructure analysis and tools

Example Change

Original	Modified

`void setImage(`
`  Image image`
`) {`
 	
`  widget->setImage(image);`
`}`
	`void setImage(`
	`  Image image`
	`) {`
	
	`  if (options) {`
	`  options->setImage(image);`
	`  }`
	`}`

srcDiff Example: XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<unit xmlns="http://www.srcML.org/srcML/src" xmlns:diff="http://www.srcML.org/srcDiff" revision="1.0.0" language="C++"
filename="old.cpp|new.cpp">
<function><type><name>void</name></type> <name>setImage</name><parameter_list>(
  <parameter><decl><type><name>Image</name></type> <name>image</name></decl></parameter>
)</parameter_list> <block>{<block_content>

<diff:insert><diff:ws>  </diff:ws><if_stmt><if>if<diff:ws> </diff:ws><condition>(<expr><name>options</name></expr>)</
condition><diff:ws> </diff:ws><block>{<block_content><diff:ws>
  </diff:ws><diff:common>  <expr_stmt><expr><call><name><name><diff:delete type="replace">widget</
diff:delete><diff:insert type="replace">options</diff:insert></name><operator>-&gt;</operator><name>setImage</name><
name><argument_list>(<argument><expr><name>image</name></expr></argument>)</argument_list></call></expr>;</expr_stmt>
</diff:common><diff:ws>  </diff:ws></block_content>}</block></if></if_stmt><diff:ws>
</diff:ws></diff:insert></block_content>}</block></function>
</unit>
```

srcDiff - Unified View

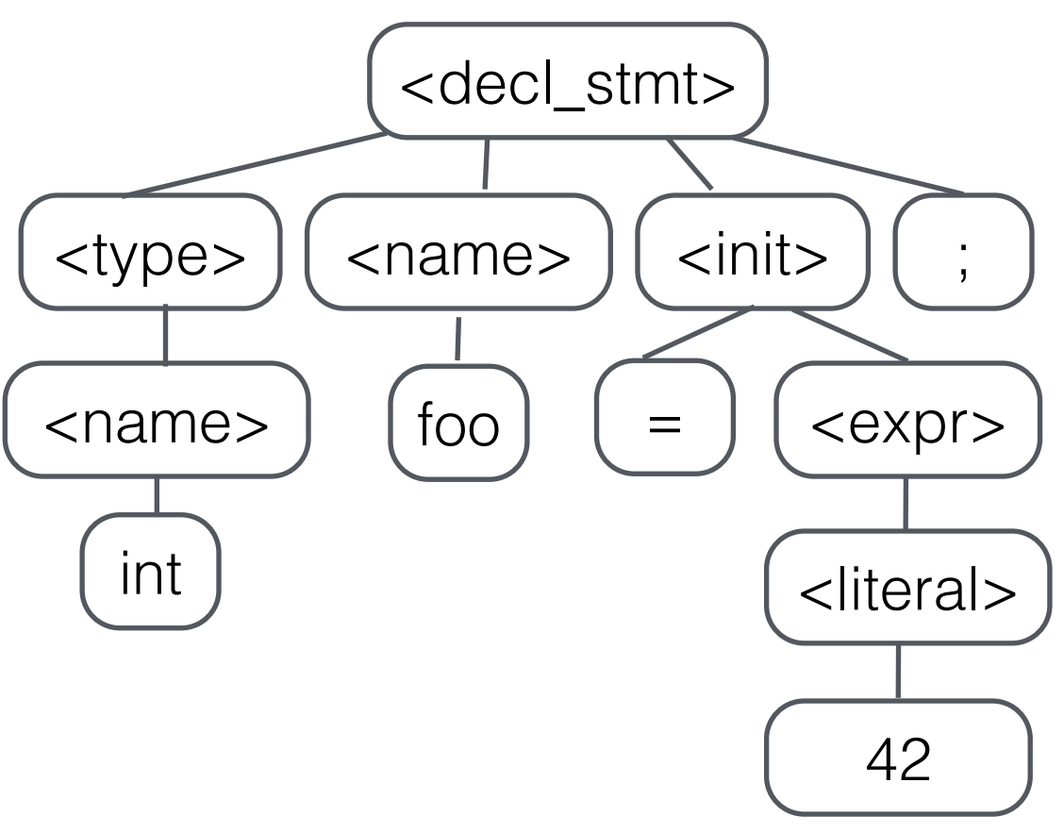
```
void setImage(
  Image image
) {

  if (options) {
    widgetoptions->setImage(image);
  }
}
```

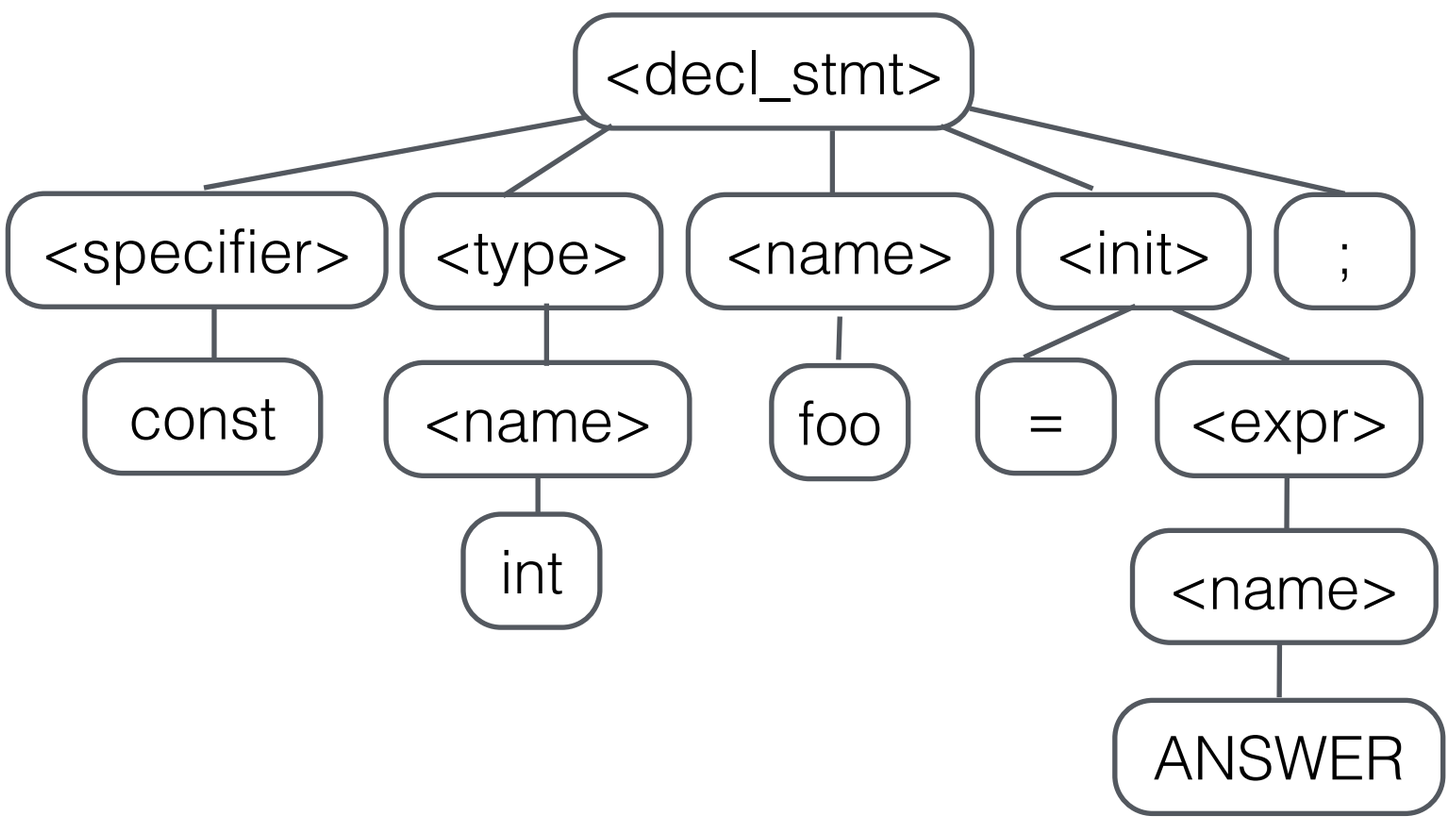
srcDiff Process

- Simultaneous preorder traversal on both the original and modified AST
- Applies a sequence differencing algorithm [Myers ’86] to the original and modified children (including subtrees) of a node
- Changed children (delete/insert same position) are analyzed for further action
  - Newer Version
  - Nested
  - Deleted or Inserted
- Actions determined by set of rules derived from how programmers change code

Process Example

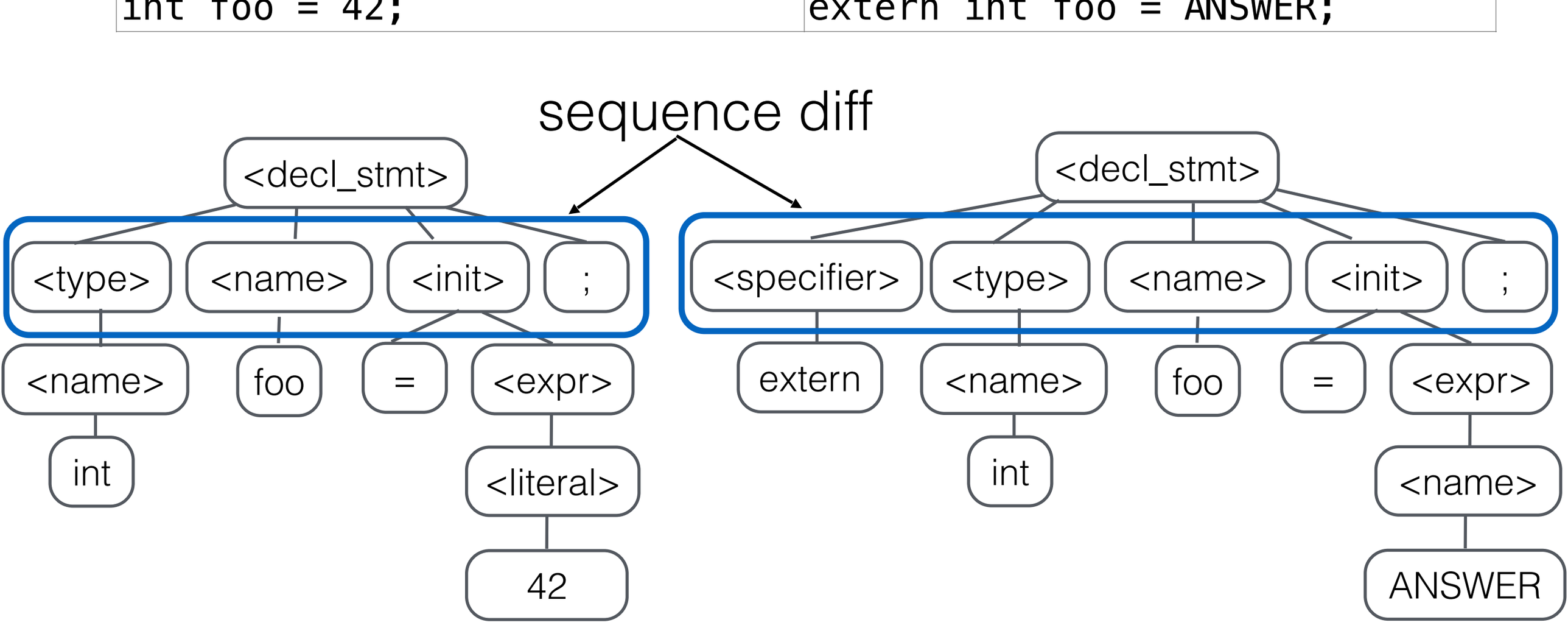
Original	Modified
`int foo = 42;`	`const int foo = ANSWER;`





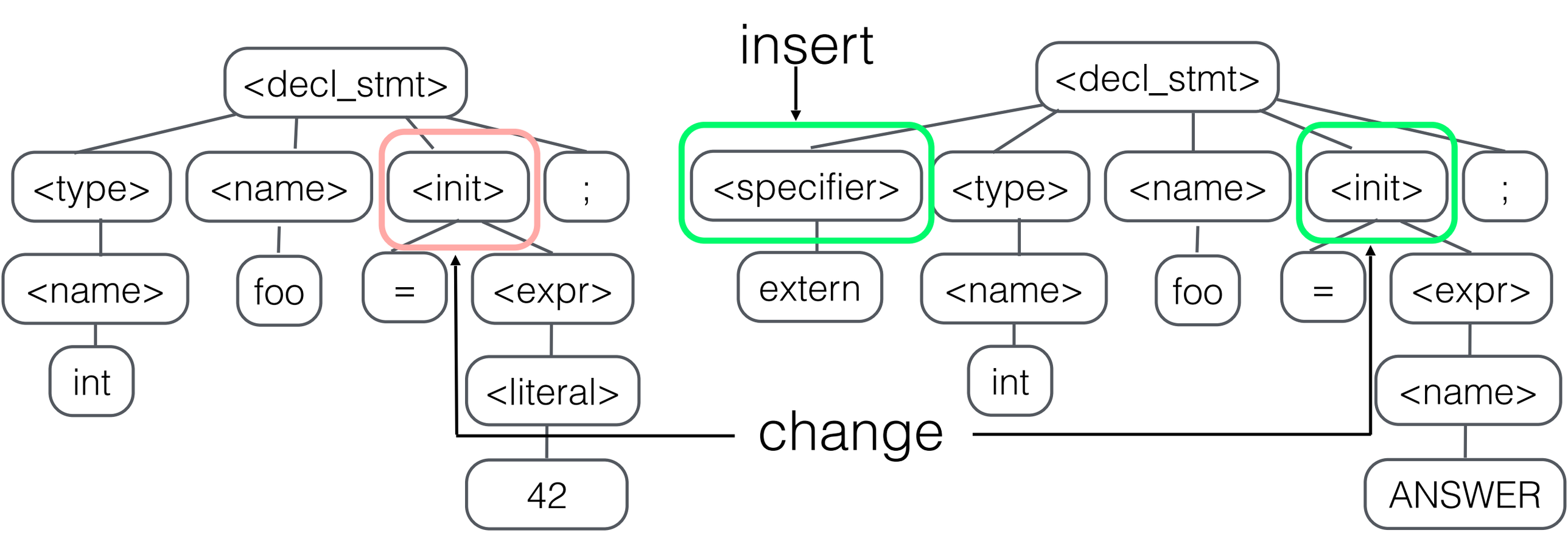
Process Example

Original	Modified
`int foo = 42;`	`extern int foo = ANSWER;`



Process Example

Original	Modified
`int foo = 42;`	`extern int foo = ANSWER;`



srcDiff Rules

- Derived from grammar of language, empirical and statical analysis of software, and experience of several expert developers
- Categories
  - Match
  - Convertibility
  - Nesting
- Set of *Similarity Rules* used by each category

Name Match

Original	Modified
`QTextDocument * m_document;`	`QTextDocumentPtr m_document;`
srcDiff	
`QTextDocument``QTextDocumentPtr` **`*`** `m_document;`	

Logical Rule (Convert)

Original

```
int itemCount = d.items.count();
for(int i = itemCount-1; i >= 0; --i) {
    Item &sbItem = d.items[i];
    if (sbItems.widget() == widget) {
        // several common lines
    }
}
```

Modified

```
int itemCount = d.items.count() - 1;
while(i >= 0) {
    Item &sbItem = d.items[i];
    if (sbItems.widget() == widget) {
        // several common lines
    }
    --i;
}
```

srcDiff

```
int itemCount = d.items.count() - 1;
forwhile(int i = itemCount-1; (i >= 0;)  --i) {
    Item &sbItem = d.items[i];
    if (sbItems.widget() == widget) {
        // several common lines
    }
    --i;
}
```

Nesting Rule

Original	Modified

`delete m_document;`	`if (m_frames.isEmpty()) {`
	`  delete m_document;`
	`}`

srcDiff	
`if (m_frames.isEmpty()) {` `  delete m_document;` **`}`**	

Team, Future, and Getting Involved

srcML/srcDiff Team

- Jonathan I. Maletic
- Michael L. Collard
- Michael J. Decker

- Joshua Behler
- Ali Al-Ramadan
- Kyle Rossi
- Bryant Whitaker
- Andrew Symonds

- Giovanni Villalobos
- Humphrey Borketey
- Noah Al-lahabi
- Brandon Scholten
- Sophia Testa

srcML Future

- XPath extension functions - `src:isVirtual()`
- srcQL (next release)
- Input/Output formats
- Wrappers - Python, Javascript, C#, Java
- Custom XML parser
- IDE Plugins
- Update feature (via hash)

srcDiff Future

- Alpha release schedule for ICMSE’24
- Releases: MacOS, Windows, Linux Distros

- Extend Language Support for srcDiff differencing engine: Java/C#
- Core srcDiff Differencing and Analysis Library
- C Language Library
- Language Bindings (e.g., Python)
- Tools supporting software development built on Core
- Plugin/configurable Deltas for Additional Languages

Getting Involved

- Visit srcML.org & srcDiff.org
- GitHub: srcML/srcML & srcDiff/srcDiff
- We have funding to support YOUR student to contribute to srcML/srcDiff (during summer or regular term)

- Tools? Wrappers? Plugins? Language Specs?

Questions