srcML a Retrospective:
The Trials and Tribulations of Building Real Software in an Academic Environment

Professor Jonathan I. Maletic
jmaletic@kent.edu
Department of Computer Science
Kent State University
Ohio, USA

SANER 2020 Keynote
**srcML** (sõrs em el), *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.

SANER 2020
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 ➔ transform ➔ srcML ➔ src
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
#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;
}
```c
#include "rotate.h"

void rotate(int& n1, int& n2, int& n3)
{
    int tn1 = n1;
    int tn2 = n2;
    int tn3 = n3;

    n1 = tn3;
    n2 = tn1;
    n3 = tn2;
}
```
srcML Markup

- All original text preserved, including white space, comments, special characters
- Syntactic structure wrapped with tags, making them addressable
- Comments marked in place
- Pre-processor statements unprocessed
Implementation

• Parsing technology in C++ with ANTLR
• Uses libxml2, libarchive, boost

• Current speed: ~92 KLOC/second
• srcML to text: ~4.5 (~1.4 compressed)

• Allows for various input sources
  • Directories, source archives (tar.gz, etc)
srcML Parser

• Custom parser based on modifications to ANTLR parser framework
• Comments and white space in a separate token stream. C-Preprocessor in a separate token stream
• Parser produces token stream with XML tags
• Highly efficient and scalable
Language Support

- C11, K&R C
- C++14, Qt extensions
- Java SE 8
- C# Standard ECMA-334
- OpenMP pragmas
<table>
<thead>
<tr>
<th>srcML Elements</th>
<th>Pluralized elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements</td>
<td><code>&lt;if_stmt&gt;, &lt;if&gt;, &lt;else&gt;, &lt;elseif&gt;, &lt;while&gt;, &lt;for&gt;, &lt;do&gt;, &lt;break&gt;, &lt;continue&gt;, &lt;return&gt;, &lt;switch&gt;, &lt;case&gt;, &lt;default&gt;, &lt;block&gt;, &lt;label&gt;, &lt;goto&gt;, &lt;empty_stmt&gt;, &lt;foreach&gt;, &lt;fixed&gt;, &lt;block&gt;, &lt;using&gt;, &lt;unsafe&gt;, &lt;assert&gt;</code></td>
</tr>
<tr>
<td>Specifiers</td>
<td><code>&lt;specifier&gt;, &lt;extern&gt;</code></td>
</tr>
<tr>
<td>Declarations, Definitions, and Initializations</td>
<td><code>&lt;decl_stmt&gt;, &lt;decl&gt;, &lt;function_decl&gt;, &lt;function&gt;, &lt;modifier&gt;, &lt;typedef&gt;, &lt;init&gt;, &lt;range&gt;, &lt;literal&gt;, &lt;lambda&gt;, &lt;using&gt;, &lt;namespace&gt;</code></td>
</tr>
<tr>
<td>Classes, Struct, Union, Enum, Interfaces</td>
<td><code>&lt;struct_decl&gt;, &lt;struct&gt;, &lt;union_decl&gt;, &lt;union&gt;, &lt;enum&gt;, &lt;class&gt;, &lt;class_decl&gt;, &lt;constructor&gt;, &lt;constructor_decl&gt;, &lt;super&gt;, &lt;destructor&gt;, &lt;annotation&gt;, &lt;extends&gt;, &lt;implements&gt;, &lt;static&gt;, &lt;protected&gt;, &lt;private&gt;, &lt;public&gt;</code></td>
</tr>
<tr>
<td>Expressions</td>
<td><code>&lt;call&gt;, &lt;name&gt;, &lt;ternary&gt;, &lt;expr&gt;, &lt;operator&gt;, &lt;argument&gt;, &lt;argument_list&gt;, &lt;parameter&gt;, &lt;parameter_list&gt;, &lt;name&gt;</code></td>
</tr>
<tr>
<td>Generics</td>
<td><code>&lt;decl&gt;, &lt;class&gt;, &lt;function&gt;, &lt;specifier&gt;, &lt;where&gt;, &lt;name&gt;, &lt;template&gt;, &lt;typename&gt;, &lt;modifier&gt;</code></td>
</tr>
<tr>
<td>Exceptions</td>
<td><code>&lt;throw&gt;, &lt;throws&gt;, &lt;try&gt;, &lt;catch&gt;, &lt;finally&gt;</code></td>
</tr>
<tr>
<td>LINQ</td>
<td><code>&lt;from&gt;, &lt;where&gt;, &lt;select&gt;, &lt;group&gt;, &lt;orderby&gt;, &lt;join&gt;, &lt;let&gt;</code></td>
</tr>
<tr>
<td>Other (C-based)</td>
<td><code>&lt;operator&gt;, &lt;sizeof&gt;, &lt;alignas&gt;, &lt;alignof&gt;, &lt;atomic&gt;, &lt;generic_selection&gt;, &lt;specifier&gt;, &lt;asm&gt;</code></td>
</tr>
<tr>
<td>Other (C#-based)</td>
<td><code>&lt;typeof&gt;, &lt;default&gt;, &lt;checked&gt;, &lt;unchecked&gt;, &lt;sizeof&gt;, &lt;attribute&gt;</code></td>
</tr>
<tr>
<td>Other (C++-based)</td>
<td><code>&lt;call&gt;, &lt;typeid&gt;, &lt;noexcept&gt;, &lt;declspec&gt;</code></td>
</tr>
<tr>
<td>Other (Java-based)</td>
<td><code>&lt;import&gt;, &lt;package&gt;, &lt;synchronized&gt;</code></td>
</tr>
</tbody>
</table>
srcML 1.0

• Client srcml with C API libsrcml

• Freeze and version srcML tags (1.0)

• Cross-linked documentation

• Multithreaded translation for large projects:
  %srcml linux-3.16.tar.xz –o linux-3.16.xml.gz
  • Macbook Air: ~7 minutes
  • Mac Pro 6 Core: ~2 minutes
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
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
- C++ preprocessor analysis
- Reverse engineering C++ template parameter constraints
srcML Team

- Michael Collard
- Drew Guarnera
- Christian Newman
- Michael Decker
- Brian Bartman
- Heather Guarnera
- Mike Weyandt
- Vlas Zyrianov
Downloads

- Over 7000 downloads of executables since 2015
srcML Road Map

- 2000: Good idea!
- 2002: IWPC’02 Prototype
- 2003: Initial Release C/C++
- 2004: IWPC’03 Better Prototype
- 2008: Applications
- 2013: Bug Fixing
- 2014: New users
- 2015: Java
- 2016: Alpha Version
- 2017-19: Documentation
- 2020: srcML^1.0.0

Development Milestones:
- 09-12: Tool Releases
- 2014: Build Management
- 2015: GitHub
- 2016: Beta Version C#
In the Beginning

• circa 2000 Memphis, TN

• Doing research on program comprehension, software evolution with some student (Andi Marcus)

• Using LSI on source code, visualization, reverse engineering

• Need to do program analysis and fact extraction

• Large code bases
Extraction via Parsing

• Must parse the source code (compiler)
• The result is an abstract syntax tree and symbol table
• Very difficult to map AST (data) back to original source code (document)
• Programmers care about code, not the AST
• Difficulty: C++, macros, templates
Our Options

• Use someone’s tool
  • May/May not work
  • May/May not be supported
• Old platform
• Hack gcc
• Build your own specialized parser
A Good Idea!

• Came up with the idea insert AST information in the form of XML markup into the source code
  
• srcML was born
  
• Just need a parser!
  
• Came up with an initial tag set, proof of concept
  
• Andi Marcus and Tony Colston
ICSE/IWPC ‘01

- CPPX, GXL, JavaML, Columbus, TXL, etc.
Fall 2001
Moved to KSU
A Prototype

• circa 2001 Kent, OH

• Met this guy (Michael Collard) who had a keen interest in document formats (XML) and differencing. Luck has it he happened to be a great developer

• Started building a prototype parser and more formal tag set

• IWPC ’02 paper (accepted as a short)

• Susan Sim - fact extractor benchmark

• DocEng’02
IWPC ‘03

• “An XML-Based Lightweight C++ Fact Extractor”
• Improved the prototype
• Huzefa Kagdi - MS Thesis (using island grammars)

Portland 2003 - Hausi dancing
srcML Road Map

IWPC’02
Prototype

IWPC’03
Better
Prototype

Initial Release
C/C++

Applications
Bug Fixing
New users
Java

Alpha
Version

Documentation

srcML^{1.0.0}

2000

2002

2003

2004

2008

09-12

2013

2014

2015

2016

17-19

2020

Good idea!

NSF Funding

MIP ICPC

GitHub

Beta
Version

C#

ABB
Funding

Docker
Tool Releases

Build Management

srcML

1.0.0

Build Management
First “Release”

• Based on the work from the IWPC’03 paper we had an initial version of srcML that could be used (by folks outside our group)

• Released “version 2” in 2004, Linux & Windows builds

• Posted this on the lab website (word of mouth)

• Early users: Giulio Antoniol, Paolo Tonella, Andy Stefik, a group at ETH (XWeaver)
Reviewer 2?

• IWPC ’02 paper - submitted as long, accepted as short
Reviewer 2?

• IWPC ’02 paper - submitted as long, accepted as short

“I didn’t think you could build it, but now I’m using it!”
Adoption

- Seriously thought about adoption (ACSE’04 with ICSE)
- Adoption of the approach (srcML format)
  - Document view (vs data view), preservation of source code
  - Lightweight markup, efficient (size, focus)
- Adoption of the parser (usability)
  - Fast, flexible, scalable, portability, robust, interoperable
Industry Interest

• circa Oct. 2005

• Got a call from a guy. Gord. VP Corporate Development at Tira Wireless (Toronto).

• Commercial license to use srcML within their product.

• Automatic porting of applications/content to various cell phone hardware (think flip phones)

• Used heavily for 3 or 4 years
Licensing

• We needed to get a bit more serious about licensing

• GPL

  • Protect the IP

  • Easily adopted by researchers, students, practitioners using it internally

• Commercial one off licensing for products
Release & Support

• We needed to get a bit more serious about releases and maintenance
• Supported both Linux and Windows at the time
• Provided executables as compile/building was difficult
• Bug reporting via email (later Google form)
srcML Road Map

- 2000: IWPC’02 Prototype
- 2002: Good idea!
- 2003: Initial Release C/C++
- 2004: IWPC’03 Better Prototype
- 2008: Applications
  - Bug Fixing
  - New users Java
- 09-12: NSF Funding
  - MIP ICPC
- 2013: Alpha Version
- 2014: GitHub Beta Version C#
- 2015: Tool Releases
- 2016: Build Management
- 17-19: Docker
- 2020: srcML^1.0.0

Special mentions:
- NSF Funding
- ABB Funding
- Good idea!
Industry Funding

- circa 2006
- Met this guy (Brian Robinson) who worked at ABB Inc. (Brian is now at Rockwell Automation)
- Gave talk at ABB (Cleveland) in 2006
- He saw the potential to use srcML for analysis tasks at ABB
- His group moved to Raleigh-Durham (from Cleveland) around that time which slowed things down
- 2008 received the first installment of 5 years of funding (~$60K/year) mainly to support srcML and associated tools
- Dave Shepherd joined ABB and used srcML in his Sando MSVS plugin.
srcML Road Map

Applications
Bug Fixing
New users
Java
Alpha
Version
Documentation
srcML^{1.0.0}

- IWPC’02
- Initial Release
- C/C++

2000

2002
- IWPC’03
- Better Prototype

2003

2004

2008
- 09-12

2013

2014

2015

2016

17-19

2020

- NSF Funding
- MIP ICPC

- GitHub
- Beta
- Version
- C#

- Docker
- Tool Releases
- Build Management

Good idea!
Leveraging srcML

- ICSM’04 - Syntactic differencing with Collard
- SET’04 - Refactoring using XSLT with Collard
- WCRE’05 - Reverse engineering UML class models with Andrew Sutton
- TEFSE’05 - Traceability with Bonita Sharif
- ICSM’06 - Reverse engineering method stereotypes with Natalia Dragan
- SCAM’06 - Factoring differences with Collard, Huzefa Kagdi
- ICSM’07 - C preprocessor analysis with Sutton
- ICSM’08 - C++ template analysis with Sutton
- ICPC’09 - Code to design traceability with Maen Hammad
- ICSM’10 - Reverse engineering class stereotypes with Dragan, Collard
- ICSM’10 - Transformations for large scale adaptive changes with Collard, Robinson
Others using srcML

- Birrer ’04 - XWeaver aspect weaver
- Binkley ’07 - Identifier analysis
- Stefik ’07 - Accessibility (for the blind)
- Hill ’07 - Program exploration
- Marcus ’08 - Metrics computation
- Tonella, Abebe ’08 - code quality
- Abebe ’09 - Source code vocabulary analysis
- Cleland-Huang ’09 - Traceability
- Jens ’09 - Quality assurance
- Corazza ’11 - Lexical information analysis
- Gethers ’12 - Information retrieval and traceability
SCAM’11

- "Lightweight Transformation and Fact Extraction with the srcML Toolkit" - Collard, Michael Decker, Maletic
- src2srcml and srcml2src (with CLI support of XPath)
- New release with Java support
- Documented tag set
- To srcML at 25 KLOC/sec and back to src at 250 KLOC/sec
- Linux kernel as test suite
- Examples of using XPath, XSLT for fact extraction and transformation problems
srcML Road Map

- 2000: Good idea!
- 2002:_IWPC’02 Prototype
- 2003: IWPC’03 Better Prototype
- 2004: Initial Release C/C++
- 2008: Applications
  - Bug Fixing
  - New users Java
- 2013: Alpha Version
- 2015: Documentation
- 2016: Tool Releases
- 17-19: Docker Build Management
- 2020: srcML^1.0.0

Key Events:
- NSF Funding
- MIP ICPC
- GitHub Beta
- Version C#
MIP

• ICPC 2013 - San Francisco with ICSE

• Received Most Influential Paper award for our IWPC 2003 (Portland) paper on srcML

• Ric Holt: “Why did srcML survive when CPPX didn’t?”
NSF CRI

• CISE Research Infrastructure

• Program specifically aimed at supporting the construction/enhancement of infrastructure to support research in computer science

• Andi Marcus: “hey dude you should write a proposal on srcML and submit it to this new NSF program, it’s perfect for you”

• Submitted in 2011 but wasn’t funded. Did get some nice reviews and suggestions

• Submitted in 2012 and got funded ($800K) - Collaboration between KSU and U Akron. CNS 13-0592/05217
srcML Road Map

2000
- IWPC’02 Prototype
2002
- IWPC’03 Better Prototype
2003
- Initial Release C/C++
2004
- Applications
- Bug Fixing
- New users Java
2008
- 09-12
- NSF Funding
- MIP ICPC
2013
- NSF Funding
- MIP ICPC
- Alpha Version
2014
- GitHub Beta Version C#
2015
- C#
2016
- Documentation
- Docker Tool Releases Build Management
2020
- srcML1.0.0

Good idea!
2014-18: The Good Years

- What did this level of funding get us?
  - Full time developer! Yeah!
- Building (usable) software in an academic setting is difficult
  - Students come and go (developer churn)
  - Objectives are not aligned with creating long lived or high quality software
- Pressures of publishing and funding vs building a tool
- Tool building is a long game (that may or may not pay off)
- The additional engineering involved does not result in publications
A Bit of Luck

• A good graduate student does not alway equate to a great developer

• All of the people involved (actual coding) just happened to be really good developers: Collard, Decker, Guarnera, Newman, Bartman, Kagdi

• Needed compiler experience so hard to find outside of your current graduate students
Developer

• Need to manage them

• Trade-off of full time developer is that they don’t get much research done and don’t make much progress on their degree

• Decker and Bartman both spend a year as full time developers and then went back to a RA position

• After two years of full time development the project had made significant progress and RAs were adequate to keep things moving
Building Real Software

- Moved to GitHub for version control and issue tracking
- Dedicated web site (srcML.org)
- Team collaboration via Discord (why not Slack?)
- CMake build system, CPack to create installers
- Docker/Docker Compose to create Linux packages, installers, and automate testing (Ubuntu, Fedora, CentOS, OpenSUSE)
- CircleCI for continuous integration
- Windows and macOS installers
Documentation

• Big expense but critical
• Language elements for each language
• Client documentation
• API documentation
• Tutorials
• Technical Briefings ICSME’14, ICSE’15
Testing

• Unit testing for client (srcml) and API (libsrcml)
• Fine-grained testing of the parser for each language
• Over 50,000 individual parser tests in 2,564 srcML archives (files)
• Stress testing on large systems (Linux kernel) across multiple platforms
srcML Road Map

- **2000**: Good idea!
- **2002**: IWPC’02 Prototype
- **2003**: Initial Release C/C++
- **2004**: IWPC’03 Better Prototype
- **2008**: Applications
  - Bug Fixing
  - New users
  - Java
- **2013**: Alpha Version
- **2014**: NSF Funding
  - MIP ICPC
- **2015**: GitHub Beta Version C#
- **2016**: Docker Tool Releases
  - Build Management
- **2020**: srcML^1.0.0

**Timeline Events**
- **2002**: 09-12
- **2014**: 17-19

**Important Milestones**
- **2000**: NSF Funding
- **2002**: ABB Funding
- **2004**: Good idea!
Tools Build on srcML
Tools (beta release)

- srcSAX - a sax2 interface and framework for using srcML - reduce barriers to adoption
- srcSlice - highly scalable forward static slicer
- srcPtr - lightweight pointer analysis tool
- srcType - static type resolution
- srcUML - Source code to UML class diagrams
- stereoCode - method/class stereotypes
Tools (no release)

- srcDiff - syntactic differencing tool
- srcQL - syntactic aware query language
- srcTL - transformation language
- srcMX - GUI for working with srcML
- Incremental call graph generator
- srcNLP parts of speech tagger for identifiers
Syntactic Differencing
srcDiff [Decker ’17]

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

- JSEP 2019
### Example Change

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>/** sets an image */</td>
<td>/** sets an image */</td>
</tr>
<tr>
<td>void setImage(</td>
<td>void setImage(</td>
</tr>
<tr>
<td>Image image</td>
<td>Image image</td>
</tr>
<tr>
<td>)</td>
<td>}</td>
</tr>
<tr>
<td>{</td>
<td>if (options) {</td>
</tr>
<tr>
<td>widget-&gt;</td>
<td>options-&gt;</td>
</tr>
<tr>
<td>setImage(</td>
<td>setImage(</td>
</tr>
<tr>
<td>image</td>
<td>image</td>
</tr>
<tr>
<td>);</td>
<td>);</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>
### Line Diff

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

### srcDiff

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

```c
int foo = 42;
```

## Modified

```c
extern int foo = ANSWER;
```

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int foo = 42;</code></td>
<td><code>extern int foo = ANSWER;</code></td>
</tr>
</tbody>
</table>

![Diagram](image)
## Process Example

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>int foo = 42;</td>
<td>extern int foo = ANSWER;</td>
</tr>
</tbody>
</table>

### Syntax Tree Differences

Original:
```
<int><decl_stmt><type>int</type><name>foo</name><init>=<expr><literal>42</literal></expr></init></decl_stmt></int>
```

Modified:
```
<specifier>extern</specifier><int><decl_stmt><type>int</type><name>foo</name><init>=<expr><name>ANSWER</name></expr></init></decl_stmt></int>
```
Process Example

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>int foo = 42;</td>
<td>extern int foo = ANSWER;</td>
</tr>
</tbody>
</table>

```
<int> foo = 42;
<decl_stmt> <type> <name> <init> <expr> <literal> ;
int foo = <expr> <literal> ;

<decl_stmt> <type> <name> <init> <expr> <name> ;
<int> foo = <expr> <name> ;

<decl_stmt> <specifier> <type> <name> <init> <expr> <name> ;
extern int foo = <expr> <name> ;
```
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

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>QTextDocument * m_document;</code></td>
<td><code>QTextDocumentPtr m_document;</code></td>
</tr>
<tr>
<td><strong>srcDiff</strong></td>
<td><strong>m_document;</strong></td>
</tr>
<tr>
<td><code>QTextDocument * m_document;</code></td>
<td></td>
</tr>
</tbody>
</table>
## Logical Rule

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
</table>
| 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
  }
} | 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

<table>
<thead>
<tr>
<th>Original</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete m_document;</td>
<td>if (m_frames.isEmpty()) {</td>
</tr>
<tr>
<td></td>
<td>delete m_document;</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>srcDiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (m_frames.isEmpty()) {</td>
</tr>
<tr>
<td>delete m_document;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
Source Code Querying
srcQL [Bartman ’17]

- Query language that is:
  - Easy to use
  - Efficient and highly scalable
  - Syntactically aware
- srcQL is loosely modeled on SQL
- Supports syntactic pattern matching with unification
- Relations for containment, partial ordering, and functional constraints

- SANER 2017
Language

FIND search-context CONTAINS pattern

• *Search Context* - the syntactic category to be searched upon as well as the return type
• *Pattern* - A pattern or XPath expression
• Also supports the operators:
  WITHIN
  FOLLOWED BY
  WHERE
  GROUP BY
  ORDER BY
  FROM
Queries - containment

• Find all functions
  FIND src:function

• Find all functions that contain a call to new
  FIND src:function CONTAINS $T = new $X

• Find all functions with a new and delete
  FIND src:function
    CONTAINS $T = new $X
    CONTAINS delete $T
Ordering

• Find all functions with a new followed by delete
  FIND src:function
  CONTAINS $T = \text{new } \$X
  FOLLOWED BY delete $T

• Find all statements that contain a call to new
  FIND src:expr_stmt CONTAINS new $X

• Find all if statements
  FIND if() { }
Complex Query

• Find all functions containing a variable that is initialized using new and is opened within an if-statement that checks it and then followed by that variable being deleted

```
FIND src:function CONTAINS $X* $I = new $T
FOLLOWED BY $I->open()
WITHIN if($I) {}
FOLLOWED BY delete $I
```
Negation - Set Operations

- Find all functions containing a variable that is initialized using new and is opened within an if-statement that checks it and then followed by that variable NOT being deleted

\[
\text{FIND src:function CONTAINS } \$X* \quad \$I = \text{new } \$T \\
\text{FOLLOWED BY } \$I->\text{open()}
\text{WITHIN } \text{if}($I) \{} \}
\]

\[\text{MINUS} \]
\[
\text{FIND src:function CONTAINS } \$X* \quad \$I = \text{new } \$T \\
\text{FOLLOWED BY } \$I->\text{open()}
\text{WITHIN } \text{if}($I) \{} \}
\text{FOLLOWED BY delete } \$I
\]
Source Transformation
srcTL [Newman ’17]

• Transformation language written for srcML
  • Natural/simple syntax
  • Uses ANTLR to generate Internal Representation (IR) which is then interpreted into a sequence of C++ calls

• Relies on static analysis for type resolution, name generation, etc.

• JSEP 2017
XML & XPath

• Query language for selecting nodes from an XML document

• Uses file-path-like notation

• Uses srcQL to address specific XML nodes
**Maintenance Task**

### Original

```c
int *ptr = new int[n];

int *ptr;
while ((ptr = new int[n]) != 0)
{
    //code
}

return new int[n];
```

### Transformed

```c
int *ptr;
try { ptr = new int[n]; }
catch (...) { ptr = 0; }

ptr = new int[n];

int *ptr;
while (true) {
    try { ptr = new int[n]; } 
    catch (...) { ptr = 0; }
    bool var1 = ptr != 0
    if(!var1){
        break;
    }
    //code
}

int *ptr;
try { ptr = new int[n]; } 
catch (...) { ptr = 0; }
return ptr;
```
Normalizing Restructurings

- Set of transformations designed to remove isomorphisms and context [Newman ’17]
- Preserve semantics (resulting code is isomorphic of original)
- Applied before a user transformation
- Applied selectively
- Similar to refactoring
Example Transformation

FIND $VAR = new $T
FROM FIND src:function
    CONTAINS $VAR = new $T
INSERT <try>
    try{
        @xpath:self::*
    }catch(...){
        $VAR = nullptr;
    }
END AFTER self::*
REMOVE self::*
# Operators

<table>
<thead>
<tr>
<th>Operations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FROM [node]</td>
<td>Select context node, always first operator</td>
</tr>
<tr>
<td>MOVE [node] [specifier]</td>
<td>Destroy at source, insert at destination</td>
</tr>
<tr>
<td>INSERT [new node] [specifier]</td>
<td>Create new subtree at location</td>
</tr>
<tr>
<td>REMOVE [node]</td>
<td>Destroy subtree rooted at node</td>
</tr>
<tr>
<td>COPY [node] [specifier]</td>
<td>Clone subtree, insert at new location</td>
</tr>
<tr>
<td>CALL [function Name]</td>
<td>Call provided function</td>
</tr>
<tr>
<td>REPLACE [node] with [node]</td>
<td>Delete original node, insert new node</td>
</tr>
</tbody>
</table>
# Location Specifiers

<table>
<thead>
<tr>
<th>Location Specifiers</th>
<th>Corresponding xpath axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEFORE [node]</td>
<td>preceding-sibling</td>
</tr>
<tr>
<td>AFTER [node]</td>
<td>following-sibling</td>
</tr>
<tr>
<td>UPTO [node]</td>
<td>preceding</td>
</tr>
<tr>
<td>DOWNTO [node]</td>
<td>following</td>
</tr>
<tr>
<td>INTO [node]</td>
<td>descendant</td>
</tr>
<tr>
<td>OUTOF [node]</td>
<td>ancestor</td>
</tr>
<tr>
<td>TOBEGIN</td>
<td>preceding-sibling::*[first()]</td>
</tr>
<tr>
<td>TOEND</td>
<td>following-sibling::*[last()]</td>
</tr>
<tr>
<td>AS [variable]</td>
<td>Store the result of an operation as a variable</td>
</tr>
</tbody>
</table>
Neat Application of srcML
• Work with Bonita Sharif
• Want to study developers (using eye trackers) in a realistic working environment (IDE)

• Eye trackers give you the screen (x, y) a person is looking at
• Researchers (i.e., grad student) must manually determine what’s at a particular (x, y)
• Forget about scrolling and switch files
srcML & iTrace

- iTrace automatically maps eye gazes to tokens
- Done using srcML in a post-processing phase
- Maps screen (x, y) to file (line, column), then to the srcML element (token) using position option
- Result is the token being viewed along with the syntactic information of the token
- Example: a name (num1) in a condition of an if-statement within the function foo-bar in file foo.cpp
srcML Road Map

- Initial Release: C/C++
- Applications: Bug Fixing, New users
- Alpha Version
- Documentation
- srcML^1.0.0

- 2003: IWPC’03, Better Prototype
- 2004: ABB Funding
- 2008: NSF Funding, MIP ICPC
- 2013: NSF Funding
- 2014: GitHub Beta Version, C#
- 2015: Tool Releases, Build Management
- 2016: Docker
- 2020: Future?
TODO List

• Continued maintenance and releases

• More language support:
  • Domain Specific Languages (DSLs)
  • Swift, Python, javascript, etc.

• Proposal to develop a parser generator
  • Given a grammar (ANTLR syntax)
  • Generate parser to srcML
srcML

noun  |  src·M·L  |  \sōrs-em-əl\  
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 : a free software application licensed under GPL.

Get srcML v1.0.0

srcML was supported in part by a grant from the National Science Foundation (CNS 13-05292/05217) and is directed by Principal Investigators Dr. Michael L. Collard and Dr. Jonathan I. Maletic. The multi-year grant (July 2013 - June 2018) was for the enhancement and maintenance of srcML. The goal is to provide a more robust research infrastructure for the exploration, analysis, and manipulation of large scale software systems.

Executables: Windows, macOS, Linux
Issue tracking/source: GitHub