Most Influential Paper Award
21st International Work Conference on Source Code Analysis & Manipulation
2021
“Lightweight Transformation and Fact Extraction with the srcML Toolkit”
Published at SCAM 2011

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

Lightweight Transformation and Fact Extraction with the srcML Toolkit

Michael L. Collard, Michael J. Decker
Department of Computer Science
The University of Akron
Akron, Ohio 44325
collard@uakron.edu, mjfd52@zips.uakron.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, Ohio 44242
jmaletic@kent.edu

Abstract—The srcML toolkit for lightweight transformation and fact-extraction of source code is described. srcML is an XML format for C/C++/Java source code. The open source toolkit that includes the source-to-srcML and srcML-to-source translators for round-trip reverse engineering is freely available. The direct use of XPath and XSLT is supported, an archive format for large projects is included, and a rich set of input and output formats through a command-line interface is available. Applying transformations and formulating queries using srcML is very convenient. Application use-cases of transformations and fact-extraction are shown and demonstrated to be practical and scalable.

Keywords-component; Source Transformation, Fact Extraction, srcML

I. INTRODUCTION

srcML\(^1\) [1-3] is an XML format for the representation of C/C++/Java source code. The representation wraps source code and extends XML to support a rich set of text-based command line tools. Users can then apply their tools to this format. However, feedback from users prompted us to include direct support for fact-extraction and transformation into the toolset via the integration of some standard XML tools. This led to a series of new features to support a rich set of input/output formats and to make the toolkit more self-contained.

Transformation and fact extraction of source code begins with the conversion of source code to the srcML format. The tool src2srcml is used to convert all of the original source code files into the srcML format. This tool is robust in that it handles unprocessed and incomplete code. The tool is very efficient with a translation speed of 25 KLOCs/sec and can handle approximately 3,000 files per minute. For example, the entire Linux kernel can be converted into the srcML format in less than seven minutes. Once in srcML, XML tools and technologies can be used for fact extraction and transformation. This includes the use of XPath and XQuery for fact extraction,
ICSM/SCAM/VISSOFT

Williamsburg Virginia
Talk & Demo
**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.
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;
}

rotate three values

```c
#include <rotate.h>

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

    // move
    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
srcML Parser

• Custom C++ 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
srcML Time Line

- IWPC’02 Prototype
- Initial Release C/C++
- Bug Fixing
- New Users
- Apps
- Alpha Version
- Docs
- 2000
- 2002
- 2003
- 2004
- 2008
- 2009
- 2011
- 2012
- 2013
- 2014
- 2015
- 2016
- 2020
- 2021

- IWPC’03
- Better Prototype
- Funding
- SCAM
- Java
- Alpha Version
- C#
- GitHub
- Beta Version
- Docs
- Docker
- Tool Releases
- CMake
- Over 7000 downloads since 2015

- Good idea!
- SCAM MIP
- MSR FC Award srcML^{1.0.0}

SCAM 2021 MIP

14
Language Support

srcML 2011

- C++
- C
- Java

srcML 2021

- C#
- C11, K&R C
- C++14, Qt extensions
- Java SE 8
- OpenMP pragmas
- LINQ
## Implementation

### srcML 2011

- Two programs: `src2srcml`, `srcml2src`
- `src` to `srcML`: ~25 KLOC/sec
- `srcML`/text: ~4.5
- `src` to `srcML` of `linux-2.6.38.3`: 7.5 minutes

### srcML 2021

- One client program: `srcml`
- Separate C-API, `libsrcml`
- `src` to `srcML`: ~250 KLOC/sec (multithreading)
- `srcML`/text: ~4.5
- `src` to `srcML` of `linux-2.6.38.3`: 1 minute
Markup Changes

**srcML 2011**

- Optional markup for operators, literals, and type modifiers
- Separate namespaces for operators, literals, and type modifiers

**srcML 2021**

- Markup of operators, literals, and type modifiers is now standard
- Operators, literals, and type modifiers now part of core elements
- Additional namespace for OpenMP pragmas
Additional Markup

• Wrap inner contents of a block:
  \[
  \text{<block>}{\text{<block_content>}}\text{</block_content>}}</block>
  \]

• Provide a pseudo block element for nested statements:
  \[
  \text{<while>}{\text{while <condition>()}}</condition>\text{<block type="pseudo">}<block_content/>\text{</block>}}</while>
  \]

• Restructure if-stmt markup to improve representation of nested if-else:
  \[
  \text{<if_stmt>}{\text{<if}>if <condition>()</condition>\text{<block>}{<block_content/>}}</block>}</if>
  \text{<if type="elseif">else if <condition>()</condition>\text{<block>}{<block_content/>}}</block>}</if>
  \text{<else>else\text{<block>}{<block_content/>}}</block>}</else>\text{</if_stmt>}
  \]
Using srcML 2011

• foo.cpp ➔ src2srcml + XPath

• foo.cpp ➔ src2srcml ➔ foo.cpp.xml ➔
  • XML Tools (e.g., XSLT, XPath)
  • application code + XML parser (e.g., libxml2, Python, etc.)
Using srcML 2021

- foo.cpp → \textit{srcml} + XPath

- foo.cpp → \textit{srcml} → foo.cpp.xml
  - \textit{XML Tools} (e.g., XSLT, XPath)
  - application code + \textit{XML parser}
  - \textit{srcSAX} framework

- foo.cpp → application code + \textit{libsrcml}
  - \textit{XML Tools} (e.g., XSLT, XPath)
  - application code + \textit{XML parser}
  - \textit{srcSAX} framework
Applications of srcML

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

- 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
srcML Team

- Michael Collard
- Jonathan Maletic
- Drew Guarnera
- Christian Newman
- Michael Decker
- Brian Bartman
- Heather Guarnera
- Mike Weyandt
- Vlas Zyrianov
TODO

• Continued maintenance and releases

• More language support:
  • Swift, Python, javascript, etc.
  • DSLs, language extensions

• Developing a parser generator
  • Given a grammar/examples (in srcML)
  • Generate parser to srcML
srcML

noun | src·M·L | ˈsœrs-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.

srcML.org - downloads
Windows
macOS
Linux - Ubuntu, Fedora, CentOS, openSUSE
GitHub - srcML
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 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.
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