Object-Oriented Programming

Concerns

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

Concerns

A concern is a particular set of information that affects the code of a computer program

- A section of conceptual information

- Want to *identify* concerns

- Want to *separate* concerns

- Separating concerns is the primary tool we have to reduce complexity and complications

Why Identify Concerns

- **Clarify Functionality** Recognizing distinct sets of information (or concerns) helps understand what each part of the system is responsible for
- **Enhance Communication** When team members understand the different concerns, collaboration is improved, and miscommunication is reduced
- **Map Requirements to Code** Identifying concerns ensures that each functional requirement is addressed explicitly in the code
- **Reveal Cross-Cutting Issues** It helps pinpoint concerns (e.g., logging, security, error handling) affecting multiple application parts
- **Prevent Overlap an** all the concerns, dev redundant code and
- **Facilitate Refactori** makes updating or r easier without unint

Why Separate Concerns

- **Improve Modularity** Separating concerns leads to a more modular architecture, making the system easier to understand and manage.
- **Enhance Maintainability** Changes can be made to one part of the system with minimal impact on others, simplifying debugging and future enhancements.
- **Boost Reusability** Isolated components are easier to reuse across different projects or within different parts of the same application.
- **Simplify Testing** With concerns separated, individual components can be tested in isolation, leading to more reliable and targeted tests.
- **Reduces Complexit** into distinct, manage overall system comp reason about
- **Enables Parallel De** on different concern interfering with one development proces
- **Aligns with Best Pra** principles like the *Si* (SRP), ensuring that purpose

srcFacts: Concerns

- srcFacts report - The source code counts
- Markdown - Markdown syntax, table format
- srcML - An XML application consisting of XML elements specifically for source code
- XML concepts - The parts of XML, and how we get them
- XML parsing - What is valid XML, how to parse at a low level

srcML & XML

- srcML is an *XML application* and one of many, e.g., XHTML

- To use srcML, you must know about the XML format and an interface to an XML parser

- However, to use XML, you **do not have to know anything about srcML**

- To use an XML parser, you **should not have to know anything about low-level XML parsing**

Count Articles

CountArticles

Count Articles: Concerns

- English-language articles

  i.e., "the", "a", "an"

- PDF content

  e.g., a PDF contains text

- PDF format

  e.g., how PDF stores content

Count Articles: Separate Concerns

- English-language articles

  → CountArticles.cpp

- PDF content

  → PDF interface declared in pdf_reader.hpp

- PDF format

  → PDF implementation in pdf_reader.cpp

Why Separate Concerns?

- Focus on one part at a time (*modularity*)
- Work with very complex formats (e.g., a PDF) or systems by only understanding the interface (*usability*)
- Usability is not just for end users
- Test separately
- Use for other purposes (*reusability*)
- Focus on *fault tolerance*, *reliability*, *performance*, *scalability*, *robustness*, *extensibility*, *maintainability*, *security*, *portability*

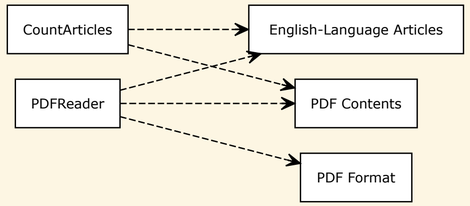
Count Articles Design

- Does the current design of CountArticles completely separate concerns?

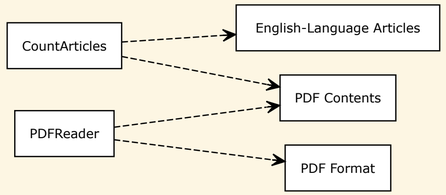
- No

- Why?

Improve Design: Separate Concerns



What We Have



What We Want

srcFacts: Primary Concerns

- srcFacts report
- Markdown
- srcML
- XML concepts
- XML parsing

Goal: Separate srcFacts Concerns

- srcFacts report → srcFacts.cpp
- Markdown → srcFacts.cpp
- srcML → srcFacts.cpp
- XML concepts → srcFacts.cpp & xml_parser.hpp
- XML parsing → xml_parser.cpp

Overall srcFacts Project Goal

- Get as much of the XML parsing details out of srcFacts.cpp as possible
- Isolate srcML to srcFacts.cpp -> No knowledge of srcML in xml_parser
- Isolate srcML to srcFacts.cpp -> No use of srcML element names in xml_parser, e.g., xml_parser **cannot** contain the srcML element name "return"
- At this point, without additional language features, unable to fully separate XML parsing from srcFacts (No, I am not just talking about classes)

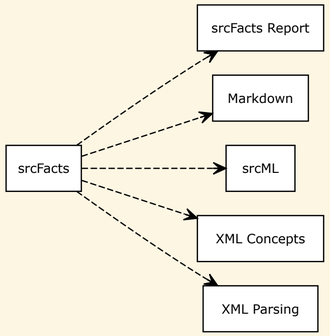
Why Separate srcFacts Concerns

- Reuse of xml_parser for non-srcML projects
- Test xml_parser (untestable in current state)
- Write other source-code reports by reusing xml_parser

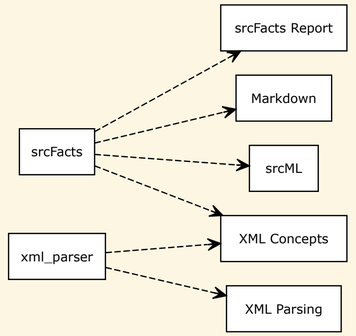
Concern Ranking

Concern	LOC (%)
XML Parsing	542 (83%)
srcFacts	55 (8%)
XML Concepts	36 (6%)
Markdown	10 (2%)
srcML	9 (1%)

srcFacts Design

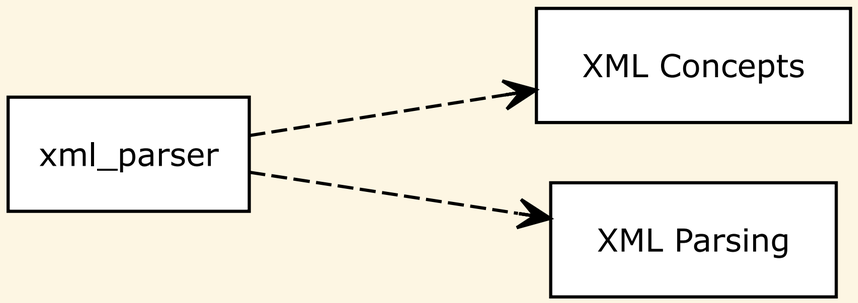


Adhoc Design



Improved Design

Separate XML Concerns



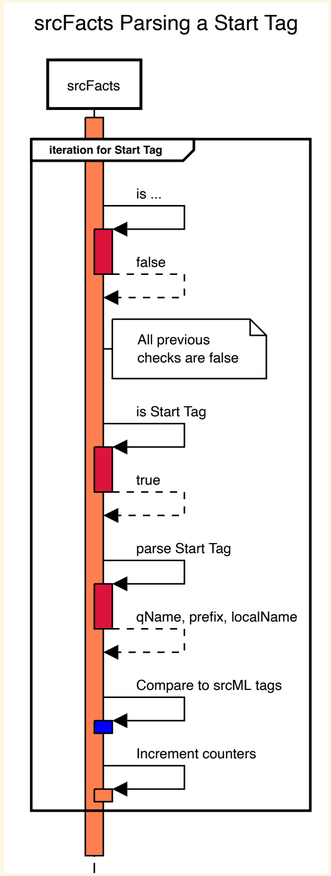
- Contains all details of the XML format
- Does not depend on srcML
- **Has no knowledge of srcML**

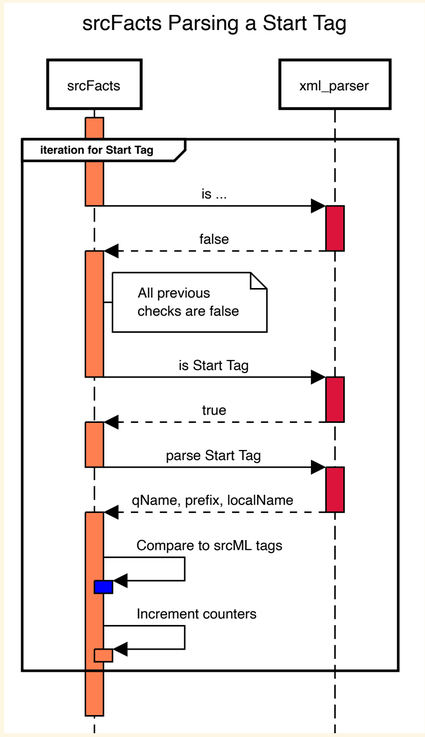
  Does **not** know that there is a srcML element "return"

  Does **not** know that there is a srcML element "literal"

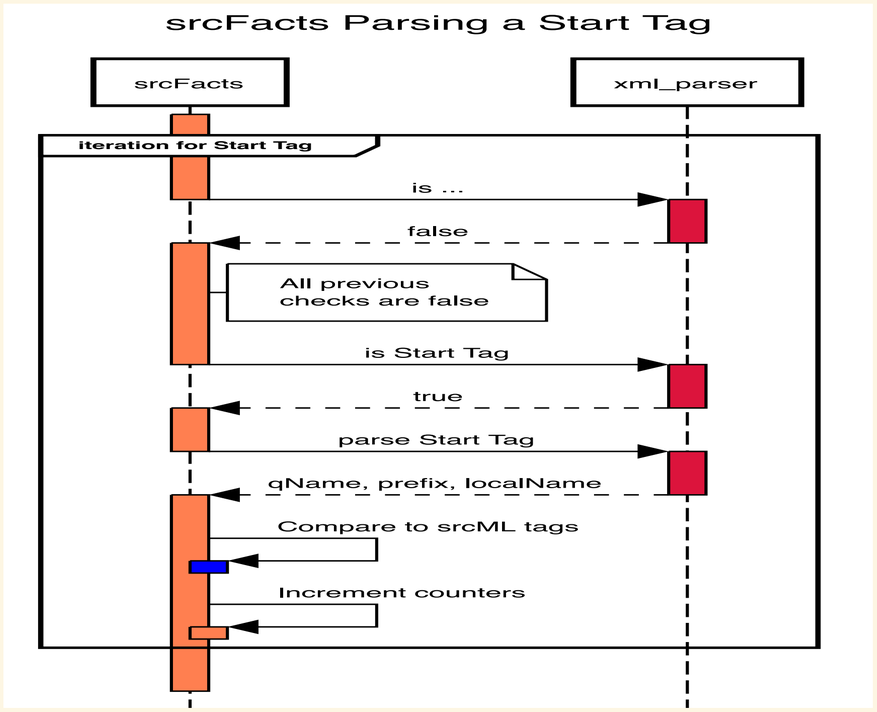
  Does **not** know that there is a srcML attribute of "type"

- Can be reused in other programs which access XML data


srcFacts Parsing a Start Tag
srcFacts
iteration for Start Tag
is ...
false
All previous
checks are false
is Start Tag
true
parse Start Tag
qName, prefix, localName
Compare to srcML tags
Increment counters


srcFacts Parsing a Start Tag
srcFacts
xml_parser
iteration for Start Tag
is ...
false
All previous
checks are false
is Start Tag
true
parse Start Tag
qName, prefix, localName
Compare to srcML tags
Increment counters

Sequence Diagrams



- srcFacts *calls* the functions in xml_parser
- Control shifts from srcFacts to xml_parser until the function *returns*
- Control goes back to srcFacts
- Comparison to srcML and incrementing counters can be viewed separately

```
if (localName == "expr"sv) {
    ++exprCount;
}

bool isExpr = localName == "expr"sv;
if (isExpr) {
    ++exprCount;
```

What is possible with a separate XML parser?

- Test parser in isolation
- Time the parser
- Replace with another XML parser
- Write lots of source reports
- Perform a source-code transformation, e.g., convert from an index-based `for` to a range-based `for`
- Use for any application which needs XML parsing