6-1-2002

XSLT and XQuery as Operator Languages

A Abram White
Dartmouth College

XSLT and XQuery as Operator Languages

A. Abram White
abewhite@mac.com

Dartmouth College Computer Science

Dartmouth College Technical Report TR2002-429

Abstract

Ubiquitous computing promises to integrate computers into our physical environment, surrounding us with applications that are able to adapt to our dynamics. Solar is a software infrastructure designed to deliver contextual information to these applications. Solar represents context data as events, and uses small programs called operators to filter, merge, aggregate, or transform event streams. This paper explores the possibility of using XSLT and XQuery to build language-neutral Solar operators.

Introduction

Solar is designed to provide information about the dynamics of the environment to context-sensitive applications. These applications may be written in a variety of computer languages executing on vastly different platforms.

XML is quickly becoming the standard method of transferring structured data between heterogeneous systems. XML parsers exist today for almost every major computer language. Therefore, Solar can gain the interoperability it needs by publishing its events in XML.

Given that Solar events may be represented in XML, it is interesting to consider whether Solar operators, whose main function is to manipulate event streams, can take advantage of languages designed to manipulate XML documents. There are two such languages worth investigating: XSLT and XQuery. To asses their utility as operator languages, each must be evaluated with respect to the following criteria:

1. Ease of use. There are three categories of operators that must be implemented – transformations, filters, and aggregations (note: mergers, the final category of operators, do not require logic). How well does the language express each of these actions, if at all?

2. Performance. Will the use of the language impose insurmountable performance overhead?

3. Maturity. Is the language and its associated technology currently robust enough to be of practical use?

XSLT as an Operator Language

1. XSLT is a language for transforming XML documents. As such, it may serve as the perfect language in which to define transformation operators. XSLT can also be used to easily create filters by transforming an input event into an empty document if it does not meet the required criteria and using an identity transformation otherwise. However, XSLT also has several drawbacks. First of all, it is extremely verbose. It is also recursive, making it more difficult for many developers to grasp than typical procedural and object-oriented scripting languages. Fortunately, though, the ease with which it manipulates XML documents generally outweighs these minor drawbacks. Of much more concern is the fact that XSLT has no built-in concept of internal state. By giving the necessary state as input along with each processed XML event and retrieving the updated state from the output of the XSLT program, it may be possible to construct a system in which state is held external to the operator itself – but in addition to being inefficient, this seems to be pushing the language far beyond its intended use. Thus stateful operators (most aggregations will fall into this category) cannot be expressed naturally. Also, complex operators may require access to services such as sockets, databases, etc; APIs for these resources are not available in XSLT.

2. Modern XSLT interpreters generally use dynamic compilation to avoid re-parsing and re-interpreting the program for each XML document it is used to process. This and other optimization techniques can greatly improve the performance of XSLT, but it will remain a concern until evaluated in practice.

3. XSLT is used in many commercial applications, and related software is widespread (especially for Java, Solar's native platform).

XQuery as an Operator Language

1. XQuery is a language for querying data sources represented in XML. It shares a common core with XSLT, and indeed the functionality of the two languages overlaps in many areas. This gives XQuery some of the strengths of XSLT, but unfortunately for the purpose of operator definition it also shares many of the same weaknesses. On a positive note, it is less verbose and more easily grasped than XSLT and improves slightly on XSLT's filtering capabilities. However, it is less well suited for transformations, and it does not solve the two major problems associated with XSLT – XQuery cannot hold state, nor can it be used to access external resources such as sockets.

2. XQuery was designed with performance in mind. However, real-world performance is unknown (see below).

3. The XQuery specification is only a working draft; it has not yet been finalized. The technology is in its infancy, and is unproven in real-world use.

Conclusion

Due primarily to their inability to hold state, neither XSLT nor XQuery seem suitable as universal operator languages. Additionally, while XQuery looks to be a promising language, the technology surrounding it is too immature for use in Solar. XSLT, however, is relatively mature, and pending real-world tests of performance and practical ease-of-use, may prove to be an excellent format for the specification of simple transformation and filter operators.