Xact – Type Safe Programming with XML
Interactive Web Applications and Services with Java and XML

Lecture 5

Troels C. Damgaard

Department of Theoretical Computer Science (ITU)

27th of September / IWSJ Fall 2005
Today, and next lecture:

A look at two (connected) technologies developed by Michael Schwartzbach and coworkers for developing safer web applications.

- Today: XACT — an XML technology for type-safe XML programming (i.e. static guarantees of output validity)
- Next lecture: JWIG — a Java technology (building on top of servlets) aiming to improve (in particular) session handling (— also employs XACT technology).

In other words, focus is on XML technology today.
OUTLINE

1 Introduction

2 XML TRANSFORMATIONS

3 XACT
   • XML Templates
   • XACT Operations
   • Example: Sorting XML
   • XACT Runtime

4 Compile-time Output Validation
   • XACT Analysis

5 Conclusion

6 Installing and running XACT

7 Exercises

8 Literature / links

9 Mini Project 1 Questions
XML as data interchange format:

- **Web services**: generate XHTML, SOAP envelopes, and customized XML documents
- **Database systems**: view and update relational data via an XML interface
- **Desktop programs**: import and export standardized XML documents

An XML transformation:
Existing solutions

Typical approaches:

- **Domain-specific** XML transformation languages (eg. XSLT)
- XML frameworks for **general-purpose languages** (eg. JDOM in Java)

Problems:

- Low-level operations
- Poor runtime performance
- Little or no check of validity

XACT goals:

- **Efficient high-level** XML transformations in a general-purpose language (Java)
- **Compile-time guarantees** that the dynamically transformed XML documents are valid according to a given XML schema
Invalid output from an XML transformation may have fatal consequences.

In order to check validity of XML documents we use XML schemas:

- An XML schema $S$ describes a set $L(S)$ of XML documents
- An XML document $X$ is valid with respect to an XML schema $S$ iff $X$ is in the set $L(S)$

In other words, an XML schema is a type for XML documents.
XML schema languages:

- DTD
- XML Schema
- DSD2
- RELAX NG

Schema processors exist for each of these technologies.
Is the output valid?

XML transformations are programs; so the question to ask is really:

Given an XML transformation $T$ and an XML schema $S$, is it the case that whenever $T$ outputs an XML document $X$, then $X$ is valid with respect to $S$?

Existing XML transformation technologies - including XSLT and JDOM - lack support for such compile-time output validation.

The consequence: Testing, testing, testing... and there might still be bugs.
1. Introduction
2. XML transformations

3. XACT
   - XML Templates
   - XACT Operations
   - Example: Sorting XML
   - XACT Runtime

4. Compile-time Output Validation
   - XACT Analysis

5. Conclusion
6. Installing and running XACT
7. Exercises
8. Literature / links
9. Mini Project 1 Questions
**XACT**

**XACT** is an API for writing XML transformations within Java.

From Java we get for free:

- The strength of a general-purpose programming language
- A rich and well-known standard library
- Platform independence
- … *(you know the drill)*

Furthermore, **XACT** includes the following combination of features:

- High-level operations for **construction** and **deconstruction** of XML documents
- XML trees as first-class Java values
- Good runtime **performance**

The main selling point:

- **XACT** provides **compile-time validation** of transformed XML documents
An XAct transformation manipulates XML templates:

- Well-formed XML fragments
- Containing named gaps (or holes)
- Immutable (aka value-based)

Construction:

- The gaps of an XML template are plugged with values

Deconstruction:

- Navigation using XPath expressions
- The addressed fragments are selected or "gapified"
An XML template is a well-formed XML fragment with named gaps. There are template gaps and attribute gaps.

(Example – next page.)
XML TEMPLATE FROM THE Recipe example

<recipe key=[key]>
  <title><![CDATA[title]]></title>
  <ingredients><![CDATA[list]]></ingredients>
  <preparation><![CDATA[steps]]></preparation>
  <comment><![CDATA[comment]]></comment>
  <![nutrition]]>
</recipe>
The plug operation:

\[
\text{plug}(g, \Delta) = \Delta
\]

\[
\text{plug}(g, \Gamma) = \Delta
\]

The values being plugged in are XML templates or strings (or arrays of these).
The `select` and `gapify` operations:

\[ \text{select}(a/b/c) = \]

\[ \text{gapify}(a/b/c, g) = \]
**XACT OPERATIONS**

**Basic Operations**

- `static XML constant(String xml)`
  (also available in sugared version: `[[ ... ]]`)
- `XML plug(String gap, XML x)`
- `XML plug(String gap, String s)`
- `XML plug(String gap, XML[] xs)`
- `XML plug(String gap, String[] ss)`
- `XML[] select(String xpath)`
- `XML[] gapify(String xpath, String g)`
**Utility Operations**

- `XML close()`
- `String text()`
- `XML cast(String dtd_url, String dtd_xmlns)`
- `XML analyze(String dtd_url, String dtd_xmlns)`
- `static XML smash(XML[] xs)`
- `static XML get(String xml_url, String dtd_url, String dtd_xmlns)`

(See the API for the complete list.)
PUTTING IT ALL TOGETHER

XACT provides a Java class library `dk.brics.xact.*`

The main class `dk.brics.xact.XML` contains:

- All the XACT operations
- A number of useful XACT macros
- Support for generating and loading constant XML documents
- Support for type-casting XML documents (using XML schemas)

The class is immutable (like Java’s value classes):

- It allows compile-time analysis of XACT transformations
- Thread-safety comes for free
Example: Sorting XML

Task: Given the collection of your mom’s favorite recipes, sort them alphabetically by their title:

- Recipe collection
- Sorted recipe collection

Both collections should be valid according the following XML schema:

- DTD for recipe collection

Write an XACT transformation that does the following:

- Deconstructs the collection into a list of recipes
- Sorts the list of recipes alphabetically by title
- Constructs a new collection from the sorted list
A RECIPE Comparator

import dk.brics.xact.*;
import java.util.*;

public class RecipeComparator implements Comparator {
    public int compare(Object o1, Object o2) {
        XML x1 = (XML)o1;
        XML x2 = (XML)o2;
        String s1 = x1.select("/recipe/title/text()")[0].text();
        String s2 = x2.select("/recipe/title/text()")[0].text();
        return s1.compareTo(s2);
    }

    public boolean equals(Object o1, Object o2) {
        XML x1 = (XML)o1;
        XML x2 = (XML)o2;
        String s1 = x1.select("/recipe/title/text()")[0].text();
        String s2 = x2.select("/recipe/title/text()")[0].text();
        return s1.equals(s2);
    }
}
Example: Sorting XML (cont.)

Sorting with XACT

```java
import dk.brics.xact.*;
import java.util.*;

public class XactSort {
    public static void main(String[] args) {
        XML.setDefaultXPathNamespace("http://www.brics.dk/ixwt/recipes");

        // Load collection from external file
        XML collection = null;
        try {
            collection =
                XML.get(args[0],
                    "http://www.brics.dk/~amoeller/talks/xact/recipes.dtd",
                    "http://www.brics.dk/ixwt/recipes");
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }

        // Remove the recipes from the collection
        XML[] recipes = collection.select("/collection/recipe");
        collection = collection.gapify("/collection/recipe", "r");

        ...
    }
}
```
Example: Sorting XML (cont.)

...  
// Sort by title and insert sorted collection
Arrays.sort(recipes, new RecipeComparator());
collection = collection.plugin("r", recipes);

// Validate and print
collection.analyze("http://www.brics.dk/~amoeller/talks/xact/recipes.dtd",
                   "http://www.brics.dk/ixwt/recipes");
System.out.println(collection);
}
The XactSort transformation shows the following advantages of XACT:

- It utilizes the high-level XACT operations for XML deconstruction
  ```java
  recipes=collection.select("/collection/recipe")
  collection=collection.gapify("/collection/recipe", "r")
  ```

- It utilizes Java’s built-in support for sorting arrays
  ```java
  Arrays.sort(recipes, new RecipeComparator())
  ```
Example: Sorting XML (cont.)

- It utilizes the high-level XACT operations for XML construction
  
  \[
  \text{collection} = \text{collection}.\text{plug}("r", \text{recipes})
  \]

- The tricky question: Is the output a valid recipe collection?
  
  \[
  \text{collection}.\text{analyze}("...\text{recipes.dtd}", "...")
  \]

- The XACT static analyzer guarantees us that this is, in fact, the case!

(XACT analyzer run on local files in Emacs. Files available upon request.)
Using JDOM instead. . .

*Sorting example using JDOM (in local Emacs - available upon request)*

Compared to the XACT version:

- Maximal flexibility, but complicated code
- No compile-time validation!
Main runtime implementation challenges:

- The `dk.brics.xact.XML` class is immutable
- All XACT operations must be efficient

A solution is described in a separate paper (see litt. ref. on last slides).
OUTLINE

1 Introduction
2 XML transformations
3 XACT
   • XML Templates
   • XACT Operations
   • Example: Sorting XML
   • XACT Runtime
4 Compile-time Output Validation
   • XACT Analysis
5 Conclusion
6 Installing and running XACT
7 Exercises
8 Literature / links
9 Mini Project 1 Questions
The question we would like answered...

Given an XML transformation $T$ and an XML schema $S$, is it the case that whenever $T$ outputs an XML document $X$, then $X$ is valid with respect to $S$?

Theorem (Rice’s Theorem (informally))

All interesting properties of the behavior of a program written in a Turing complete language are undecidable.

Theoretical conclusion: Compile-time output validation is impossible!

Pragmatic answer: Give approximate answers!
Instead of giving up, compiler writers resort to static analysis:

- Don’t try to decide the question exactly
- Settle for an approximative answer
- Only give safe (≃ conservative/sound) answers

For the Halting Problem, the answers would be:

- "Yes, your program definitely always terminates"
- "I don’t think your program always terminates, but I’m not really sure"

The engineering challenge is to give useful answers as often as possible - and be efficient.
The basic idea:
- Use static analysis to validate XACT transformations

Given an XACT transformation and a DTD schema, the possible answers are:
- "The output is always valid with respect to the schema"
- "The output might be invalid with respect to the schema, but I am not sure"

The main challenges are:
- Extract the XACT control-flow from the Java program
- Define a suitable abstraction of XML template values
- Define data-flow equations modelling the XACT operations

This analysis heavily exploits the immutability.
A (normalized) summary graph represents a set of XML template values:

- The analysis computes a summary graph for each expression in the program.
Non-trivial tasks:

- Construct summary graphs from XML template constants
- Construct summary graphs from DTD schemas
- Evaluate XPath expressions (symbolically) on summary graphs
- Validate summary graphs against DTD schemas
- Model data-flow by (quite complex) equations for each XACT operation
Here’s an example of a nice data-flow equation...:

\[
\Delta(\text{exp.gapify}(p, g)) =
\]
\[
(R,
T \setminus \{(n, h, m) \in T \mid m \in \text{ALL}\}
\cup \{(n, h, t) \mid (n, h, m) \in T \land m \in \text{HITS}\},
S[n \mapsto \emptyset \text{ for each } n \in \text{ALL} \cap (N_c \cup N_A)],
P'[g \mapsto (\text{open}(P(g)) \cup \{t\} \cup \text{HITS} \cap N_A),
\text{removed}(P(g)),
\text{merge}(\text{ANY}_{N_c \cup N_c}, \text{tgaps}(P(g))),
\text{merge}(\text{ANY}_{N_A}, \text{agaps}(P(g)))))
\]
import dk.brics.xact.*;

public class Hello {
    public static final String XHTML_NAMESPACE = "http://www.w3.org/1999/xhtml";
    public static final String[] NAMESPACES = {
        "h:=http://www.w3.org/1999/xhtml",
    };
    static {
        XML.setDefaultConstantNamespace(XHTML_NAMESPACE);
        XML.setNamespaceMap(NAMESPACES);
    }

    static XML person =
        XML.constant("<person><name>John Doe</name></person>");

    static XML html = XML.constant("<html>
        <head><title>Hello</title></head>
        <body bgcolor="red">
            <h1>Hello [name]</h1>
        </body>
    </html>");

    ...
Detecting Errors (cont.)

Example Program (No Errors) (cont.)

```java
public XML sayHello() {
    XML name =
        person.select("/h:person/h:name/text()")[0];
    return html.plug("name", name);
}

public static void main(String[] args) {
    Hello hello = new Hello();
    XML greeting = hello.sayHello();
    greeting.analyze("file:xhtml1-transitional.dtd","http://www.w3.org/1999/xhtml";
    System.out.println(greeting);
}
```
Let’s try an online demonstration of the XACT analyzer.

On local files in Emacs: Running the XACT analyzer introducing (at least) the following errors

- bgcolor $\mapsto$ gbcolor
- name $\mapsto$ nema (gap name)
- name $\mapsto$ name (in XPath exp)
Outline

1. Introduction
2. XML transformations
3. XACT
   - XML Templates
   - XACT Operations
   - Example: Sorting XML
   - XACT Runtime
4. Compile-time Output Validation
   - XACT Analysis
5. Conclusion
6. Installing and running XACT
7. Exercises
8. Literature / links
9. Mini Project 1 Questions
XACT is an alternative to e.g. XSLT and JDOM:

- high-level operations (based on templates and XPath)
- fully integrated into Java
- efficient runtime representation
- transformations can be validated at compile-time
1. Introduction
2. XML transformations
3. XACT
   - XML Templates
   - XACT Operations
   - Example: Sorting XML
   - XACT Runtime
4. Compile-time Output Validation
   - XACT Analysis
5. Conclusion
6. Installing and running XACT
7. Exercises
8. Literature / links
9. Mini Project 1 Questions
Outline

1. Introduction
2. XML transformations
3. XACT
   - XML Templates
   - XACT Operations
   - Example: Sorting XML
   - XACT Runtime
4. Compile-time Output Validation
   - XACT Analysis
5. Conclusion
6. Installing and running XACT
7. Exercises
   - Literature / links
8. Mini Project 1 Questions
Introduction

XML transformations

XACT
- XML Templates
- XACT Operations
- Example: Sorting XML
- XACT Runtime

Compile-time Output Validation
- XACT Analysis

Conclusion

Installing and running XACT

Exercises

LITTERATURE / LINKS

Mini Project 1 Questions
These slides are an adapted version of the presentation slides by Anders Møller for XACT available via the XACT website at http://www.brics.dk/~amoeller/talks/xact/.

More information - including a prototype implementation: http://www.brics.dk/Xact/

The publications –


– focus respectively on the theory of static analysis (based on earlier results), and on developing a runtime results.

Though somewhat technical (in particular, the former), they are recommended as supplementary reading.
The XACT website contains source code together with an installation guide for Linux (using Java 2 SE 1.4.2). For installation on Windows and/or using Java 2 SE 1.5 – see

Introduction

XML transformations

XACT
- XML Templates
- XACT Operations
- Example: Sorting XML
- XACT Runtime

Compile-time Output Validation
- XACT Analysis

Conclusion

Installing and running XACT

Exercises

Literature / links

Mini Project 1 Questions
Shoot!’