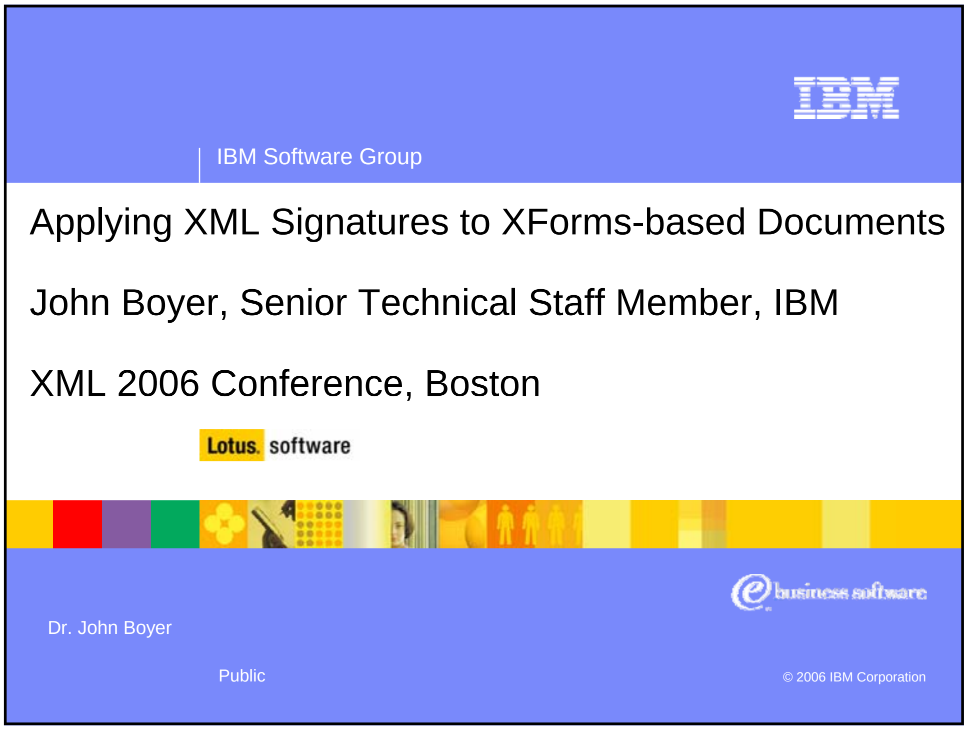

IBM Software Group

Applying XML Signatures to XForms-based Documents

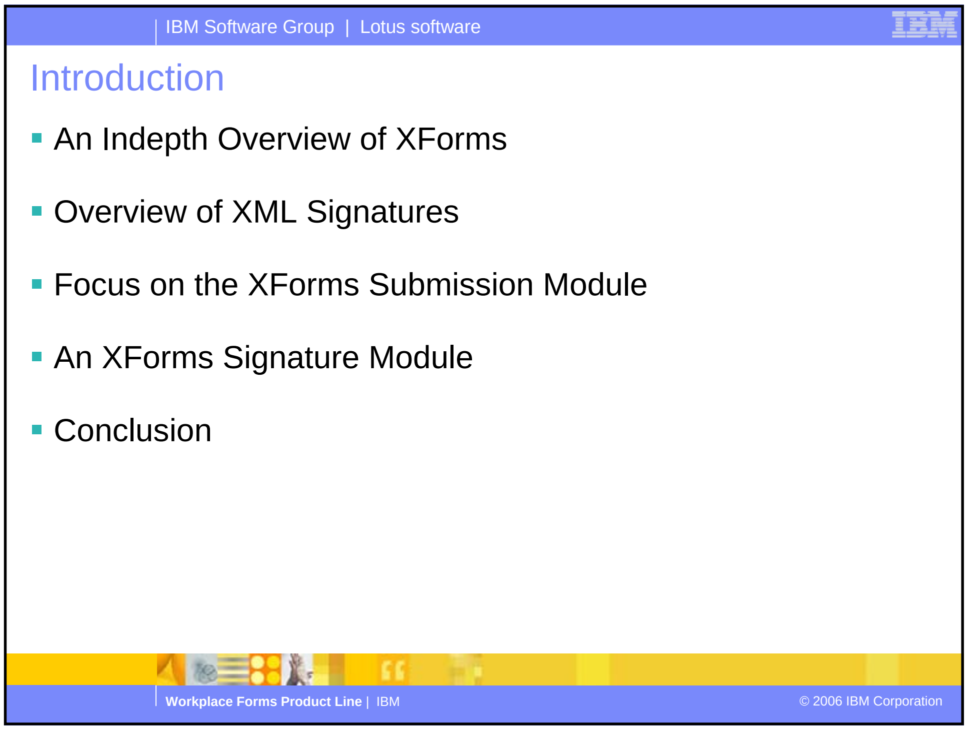
John Boyer, Senior Technical Staff Member, IBM

XML 2006 Conference, Boston

Lotus. software

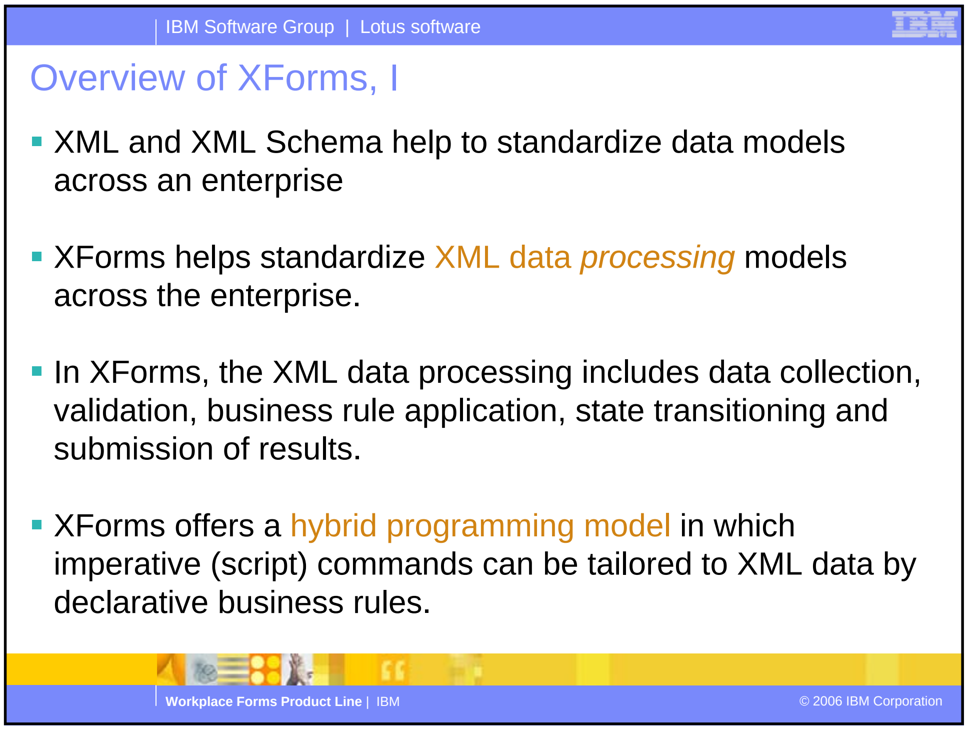
Dr. John Boyer

Public



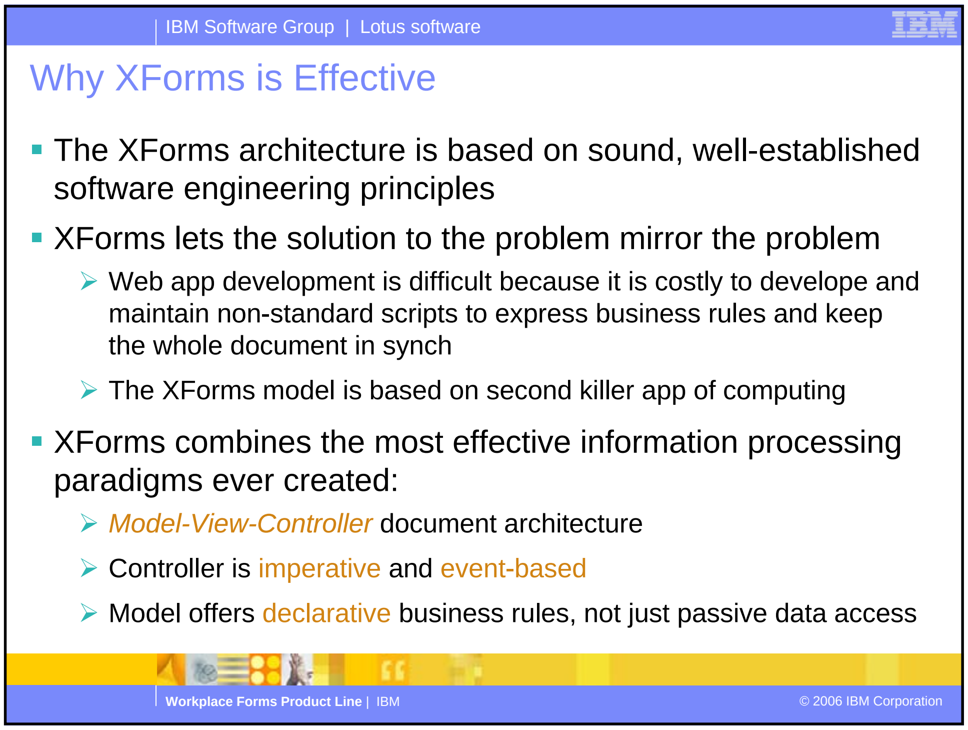
Introduction

- An Indepth Overview of XForms
- Overview of XML Signatures
- Focus on the XForms Submission Module
- An XForms Signature Module
- Conclusion



Overview of XForms, I

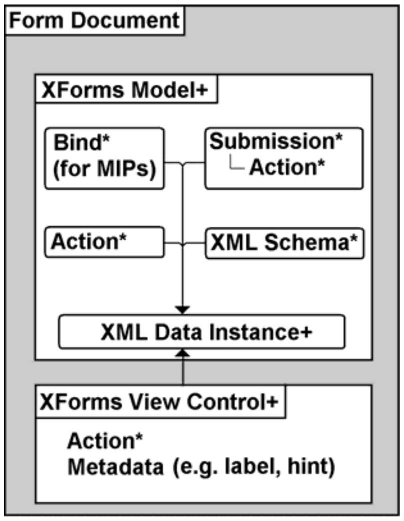
- XML and XML Schema help to standardize data models across an enterprise
- XForms helps standardize XML data *processing* models across the enterprise.
- In XForms, the XML data processing includes data collection, validation, business rule application, state transitioning and submission of results.
- XForms offers a hybrid programming model in which imperative (script) commands can be tailored to XML data by declarative business rules.



Why XForms is Effective

- The XForms architecture is based on sound, well-established software engineering principles
- XForms lets the solution to the problem mirror the problem
  - Web app development is difficult because it is costly to develope and maintain non-standard scripts to express business rules and keep the whole document in synch
  - The XForms model is based on second killer app of computing
- XForms combines the most effective information processing paradigms ever created:
  - *Model-View-Controller* document architecture
  - Controller is imperative and event-based
  - Model offers declarative business rules, not just passive data access

Internal Architecture of an XForms Document



Opening the Black Box of an XForms Document, I

- The “grey matter” represents the presentation layer markup, things like styling for fonts, colors, etc.
- Usually one model in a form document, but sometimes multiple models can appear in one document (e.g. multiple portlets running forms in a portal).
- One or more instances of XML data in a namespace and according to a schema or design chosen by the form author.
- Instances can represent records within a transaction as well as ‘local variables’ that contain intermediate results.
- Optionally, XML schema for the data may be used, either directly inline within or externally referenced by the model.
- Actions within the model allow the form author to participate in model initialization by handling events like **`xforms-ready`** and **`xforms-model-construct-done`**

Opening the Black Box of an XForms Document, II

- The bind elements implement the business rules
- These can be the not just calculated results like the total of a purchase order but also *model item properties* (MIPs) that are associated with the nodes of instance data
- A simple example of a MIP is *type*, which assigns a datatype to a node in combination with or alternatively to use of a schema
- A *constraint* provides a dynamic or data-dependent validity rule
- The *relevant* MIP corresponds to a schema choice. For example, the data for a UK address block can be made non-relevant if the user's meatspace location is in the US.
- The *readonly* MIP controls whether or not a data node can be modified during processing.
- The constraint, relevant and readonly MIPs are XPath formulae that can be dynamically changed during run-time.
- Both readonly and relevant control subtrees, not just nodes.

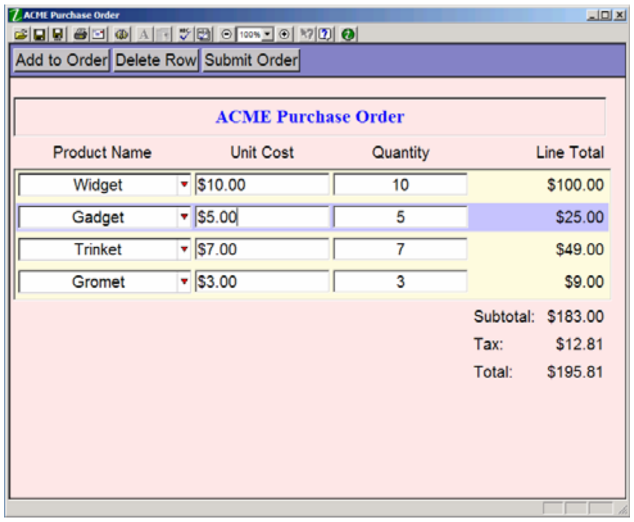
Opening the Black Box of an XForms Document, III

- There can be any number of submission elements to control information exchange between an XForm and servers
- An XForm can call a *web service* during run-time to enrich the form fill experience with information from external data sources
- When the user completes the fill experience, the data can be submitted to the next stage of the workflow, and the returned result can replace the entire document.
- The Actions within a submission allow the form author to participate in the submission process.
- A handler for the **`xforms-submit`** event allows for preparing a submission, e.g. copying raw data into the payload area of a SOAP envelope.
- The **`xforms-submit-done`** and **`xforms-submit-error`** events allow handlers to post-process the submission.

Opening the Black Box of an XForms Document, IV

- The XForms view controls expose a data interaction layer to the user.
- Control types include atomic user interface elements like *input*, *secret*, *textarea*, *select1*, *select*, and *range*.
- The *output* control provides a non-modiafiable view of data as well as a data aggregation capability.
- There are also controls that provide the ability to specify hierarchy, choice and iteration within the user interface. These are the *group*, *switch* and *repeat* elements.
- The controls bind directly to data instance nodes within the model, and they consume the content and MIPs of the nodes.
- The Actions within a control allow handlers to respond to a plethora of notification events.
- Metadata for controls includes labels, help, hints and alerts

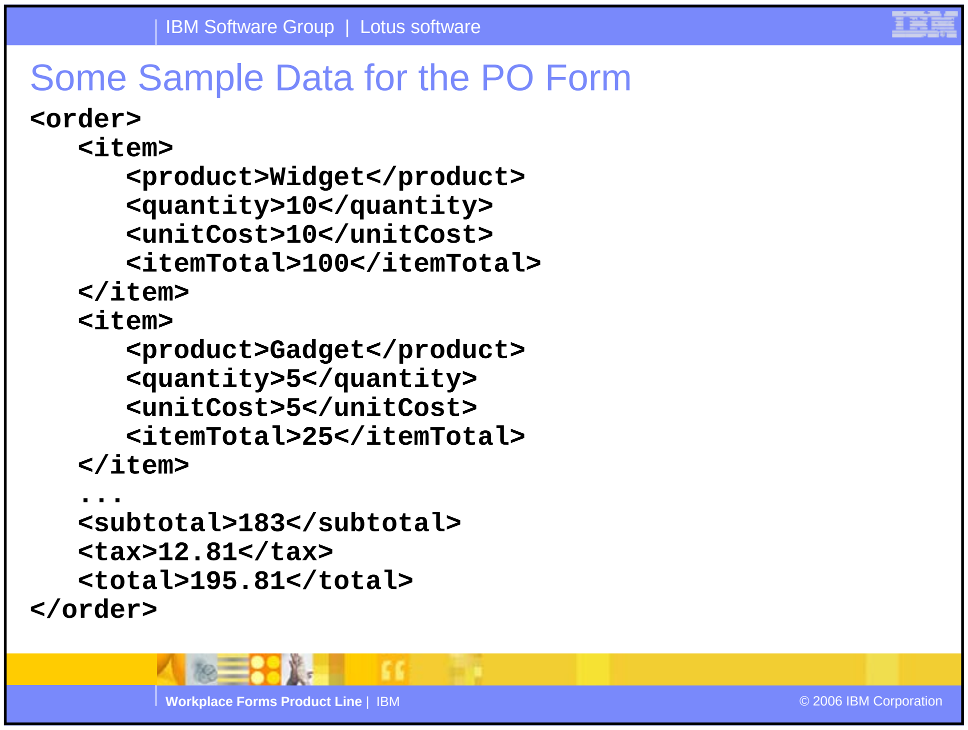
Effectiveness of the Hybrid Programming Model, I



ACME Purchase Order

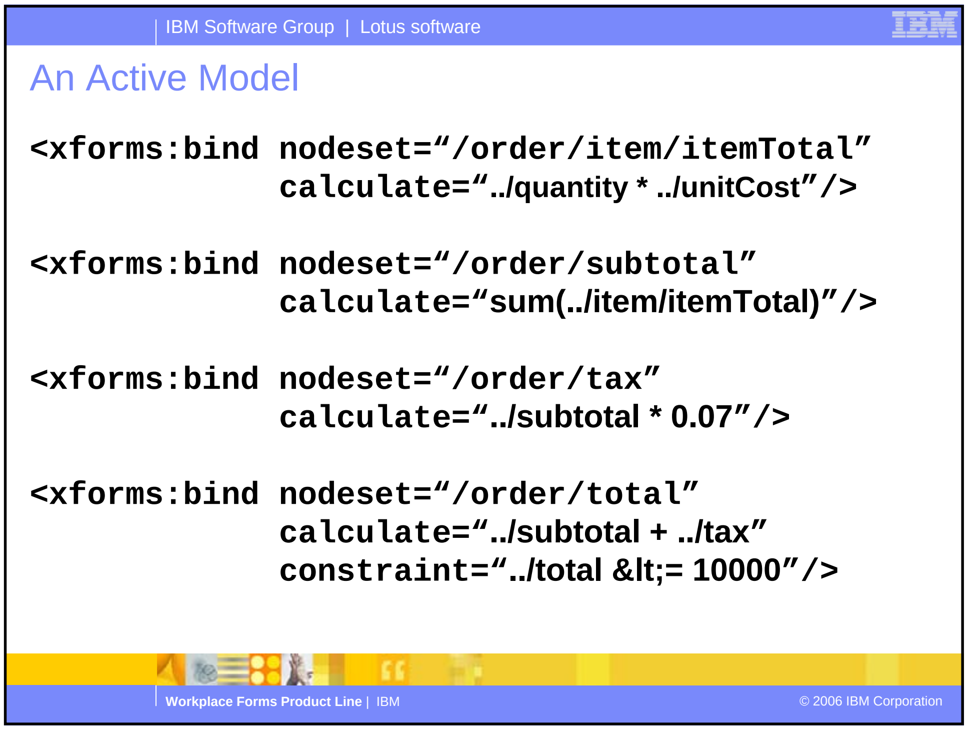
Product Name	Unit Cost	Quantity	Line Total
Widget	$10.00	10	$100.00
Gadget	$5.00	5	$25.00
Trinket	$7.00	7	$49.00
Gromet	$3.00	3	$9.00

Subtotal:	$183.00
Tax:	$12.81
Total:	$195.81



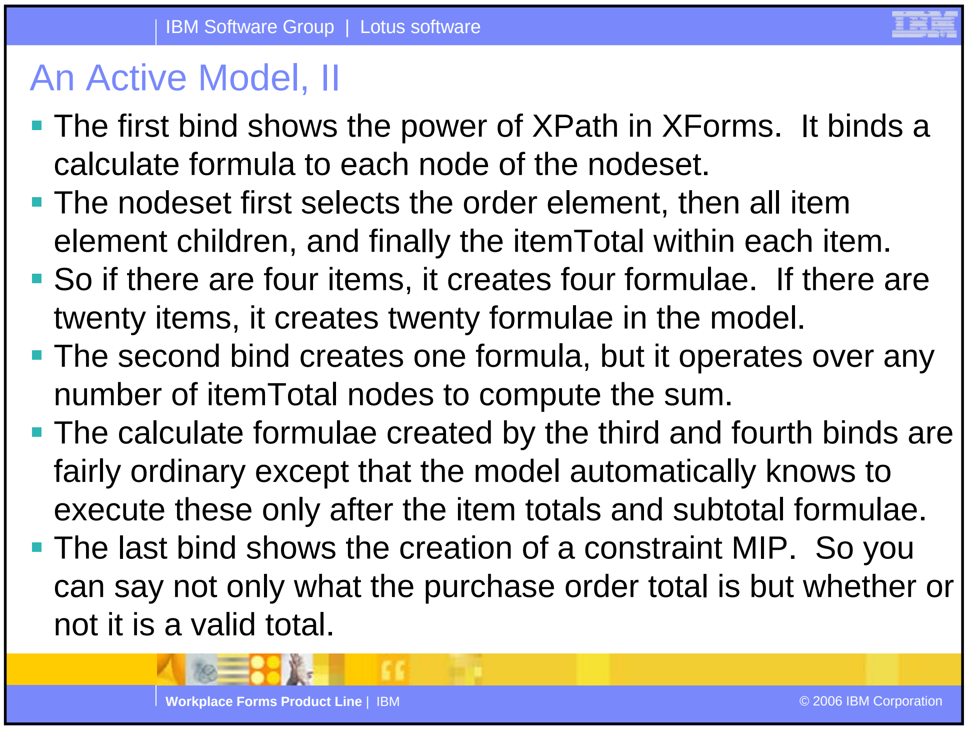
Some Sample Data for the PO Form

```
<order>
   <item>
      <product>Widget</product>
      <quantity>10</quantity>
      <unitCost>10</unitCost>
      <itemTotal>100</itemTotal>
   </item>
   <item>
      <product>Gadget</product>
      <quantity>5</quantity>
      <unitCost>5</unitCost>
      <itemTotal>25</itemTotal>
   </item>
   ...
   <subtotal>183</subtotal>
   <tax>12.81</tax>
   <total>195.81</total>
</order>
```



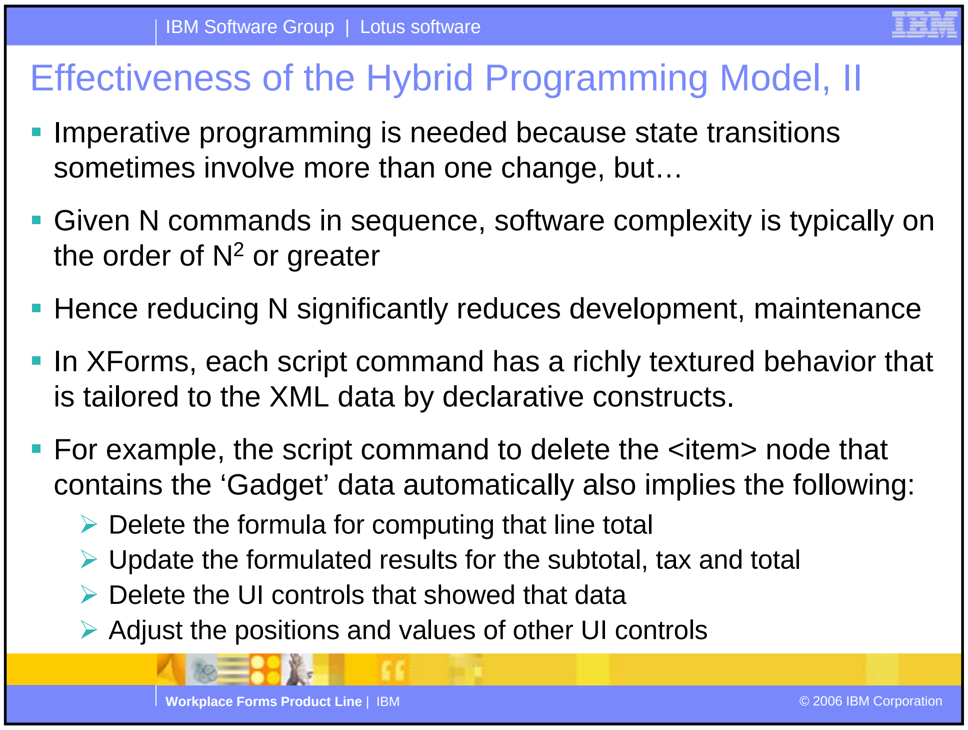
An Active Model

```
<xforms:bind nodeset="/order/item/itemTotal"
             calculate="../quantity * ../unitCost"/>

<xforms:bind nodeset="/order/subtotal"
             calculate="sum(../item/itemTotal)"/>

<xforms:bind nodeset="/order/tax"
             calculate="../subtotal * 0.07"/>

<xforms:bind nodeset="/order/total"
             calculate="../subtotal + ../tax"
             constraint="../total &lt;= 10000"/>
```



An Active Model, II

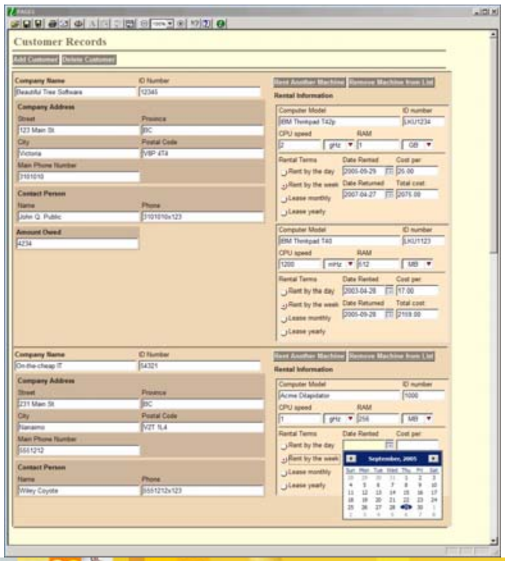
- The first bind shows the power of XPath in XForms. It binds a calculate formula to each node of the nodeset.
- The nodeset first selects the order element, then all item element children, and finally the itemTotal within each item.
- So if there are four items, it creates four formulae. If there are twenty items, it creates twenty formulae in the model.
- The second bind creates one formula, but it operates over any number of itemTotal nodes to compute the sum.
- The calculate formulae created by the third and fourth binds are fairly ordinary except that the model automatically knows to execute these only after the item totals and subtotal formulae.
- The last bind shows the creation of a constraint MIP. So you can say not only what the purchase order total is but whether or not it is a valid total.



Effectiveness of the Hybrid Programming Model, II

- Imperative programming is needed because state transitions sometimes involve more than one change, but…
- Given N commands in sequence, software complexity is typically on the order of N^2 or greater
- Hence reducing N significantly reduces development, maintenance
- In XForms, each script command has a richly textured behavior that is tailored to the XML data by declarative constructs.
- For example, the script command to delete the <item> node that contains the 'Gadget' data automatically also implies the following:
  - Delete the formula for computing that line total
  - Update the formulated results for the subtotal, tax and total
  - Delete the UI controls that showed that data
  - Adjust the positions and values of other UI controls

Imperative-Declarative Blend Scales to any Depth



Customer Records

Company Name	Beautiful Tree Software
ID Number	12345
Street	123 Main St
Province	BC
City	Victoria
Postal Code	V8P 4T4
Main Phone Number	3101010
Contact Person Name	John Q. Public
Phone	3101010x123
Amount Owed	4234

Rental Information

Computer Model	IBM Thinkpad T42p
ID number	LKU1234
CPU speed	2 gHz
RAM	1 GB
Date Rented	2005-09-29
Cost per	25.00
Date Returned	2007-04-27
Total cost	2075.00

Computer Model	IBM Thinkpad T40
ID number	LKU1123
CPU speed	1200 mHz
RAM	512 MB
Date Rented	2003-04-28
Cost per	17.00
Date Returned	2005-09-28
Total cost	2159.00

Rental Terms: Rent by the day / Rent by the week / Lease monthly / Lease yearly

Company Name	On-the-cheap IT
ID Number	54321
Street	231 Main St
Province	BC
City	Nanaimo
Postal Code	V2T 1L4
Main Phone Number	5551212
Contact Person Name	Wiley Coyote
Phone	5551212x123

Rental Information

Computer Model	Acme Dilapidator
ID number	1000
CPU speed	1 gHz
RAM	256 MB
Date Rented	
Cost per	

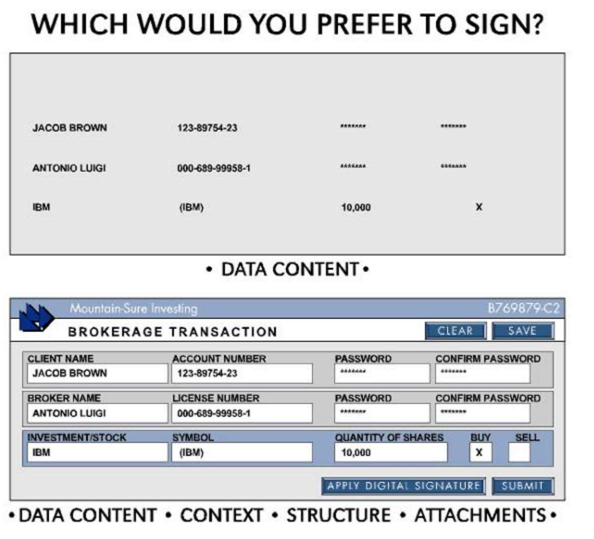
Script is *Richly* Textured by Declarative Constructs

- Each row of a repeat need not be a simple linear progression of a few controls.
- A repeat helps to provide a view for data whose schema includes elements with maxoccurs > 1, regardless of the structure of those elements
- So, the repeated user interface can have arbitrary structure and layout to match the presentational needs of the data.
- And script commands are tailored to the data by the declarative constructs. For example, delete has the same meaning in the context of a more elaborate data structure:
  - Delete all descendant nodes
  - Delete all attached UI controls (affecting flow layout as expected)
  - Delete all formulate attached to all deleted nodes
  - Update all formulae that depend on deleted nodes

Dynamic Archival Formats, Digital Signature Security

- XForms has a data-centric processing paradigm.
- The automatic behaviors created by declarative constructs are based on mutations of data performed by the commands
- The entire application state can therefore be represented by a serialization of the XML data and its original document.
- XForms solves key problem of preserving application dynamism in a document format suitable for archiving and transaction auditability.
- Implication for digital signature security is that the XML can be used to implement "What you see is what you sign"

Must sign the data, the XForm, and the Presentation



WHICH WOULD YOU PREFER TO SIGN?

JACOB BROWN	123-89754-23	*******	*******
ANTONIO LUIGI	000-689-99958-1	*******	*******
IBM	(IBM)	10,000	X

• DATA CONTENT •

Mountain-Sure Investing B769879C2

BROKERAGE TRANSACTION

CLIENT NAME	ACCOUNT NUMBER	PASSWORD	CONFIRM PASSWORD
JACOB BROWN	123-89754-23	*******	*******

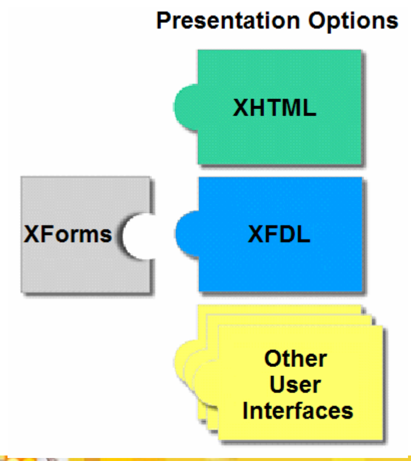
BROKER NAME	LICENSE NUMBER	PASSWORD	CONFIRM PASSWORD
ANTONIO LUIGI	000-689-99958-1	*******	*******

INVESTMENT/STOCK	SYMBOL	QUANTITY OF SHARES	BUY	SELL
IBM	(IBM)	10,000	X	

• DATA CONTENT • CONTEXT • STRUCTURE • ATTACHMENTS •

But XForms Separates ‘Purpose’ from ‘Presentation’

The core data processing asset can be ‘skinned’ by varied host languages for presentational multimodality.



Example: XFDL as a Skin for XForms

```
<field sid=“GrossIncome”>
   <xforms:input ref=“/income/gross”>
      <xforms:label>Gross Income:</xforms:label>
   </xforms:input>
   <format>
      <datatype>currency</datatype>
   </format>
</field>
...
<label sid=“IncomeTax”>
   <xforms:output ref=“/income/tax”/>
   <fontcolor compute=“ value >= ‘0’ ? ‘black’ : ‘red’ ”/>
   ...
</label>
```

For more info, see XML 2005 Conference Proceedings

Example: XFDL as a Skin for XForms, II

- The host language (XFDL) can augment the XForms UI controls with internationalized capability. For example, the raw data can conform to xsd:decimal yet be presented with currency decorations appropriate to the locale.
- The host language can provide presentation properties such as the kind of UI element (e.g. popup vs. list box), the layout position and size, fonts, colors, etc.
- The host language can provide dynamic presentational behaviors, such as automatically changing the font color between red and black based on whether there is a balance owing or balance due.
- For further details about the XFDL host language for XForms, see http://www.idealliance.org/proceedings/xml05/abstracts/paper74.HTML

Overview of XML Signatures, I

Multiple references are used to secure all resources contributing to the on-the-glass presentation of the data:

```
<Signature ID? xmlns=“&dsig;”>
   <SignedInfo>
      <CanonicalizationMethod>
      <SignatureMethod>
      (<Reference URI? >
          (<Transforms>)?
          <DigestMethod>
          <DigestValue>
       </Reference>)+
   </SignedInfo>
   <SignatureValue>
   (<KeyInfo>)?
   (<Object ID? >)*
</Signature>
```

Overview of XML Signatures, II

- Generating a signature is a two-stage process.
- Working backwards, the <SignatureValue> takes the result of the cryptographic operation involving the signer's private key
- The <KeyInfo> element stores signature metadata like the <KeyName> and possibly the public key certificate.
- <Object> tags may be used to store additional information related to the <Signature>
- However, the cryptographic operation involving the signer's private key is performed only on the <SignedInfo>
- The actual resources being secured by the signature are represented by <Reference> elements within <SignedInfo>

Overview of XML Signatures, III

- For each <Reference>, the indicated resource is transformed and digested, and the result is stored in the <DigestValue>.
- The <References> are used to secure all the resources needed to capture the full context of the XML data
- External resources like the image of a corporate logo can be referred to by a non-empty URI.
- An empty URI allows a *same document* reference, and an absent URI is for a reference to an application-defined context.
- After all the resources are secured by <Reference>'s, then the digest of the <SignedInfo> is computed, encrypted with the signer's private key and stored in the <SignatureValue>.

Submission, or How to Add a Module to XForms

```
<model xmlns=“&xforms;”>
  <instance id=“data”>
     Initial XML data
  </instance>
  <instance id=“moredata”>
     More XML data, which may be obtained at run-time
  </instance>
  …
  <submission id=“search”
     ref=“data to submit, e.g. data”
     action=“http://where to send data”
     instance=“where to put result, e.g. moredata” ...>
     ...
  </submission>
</model>
...
<xforms:trigger>
   <xforms:label>Search</xforms:label>
   <xforms:send ev:event=“DOMActivate”
                submission=“search”/>
</xforms:trigger>
```

Focus on XForms Submission

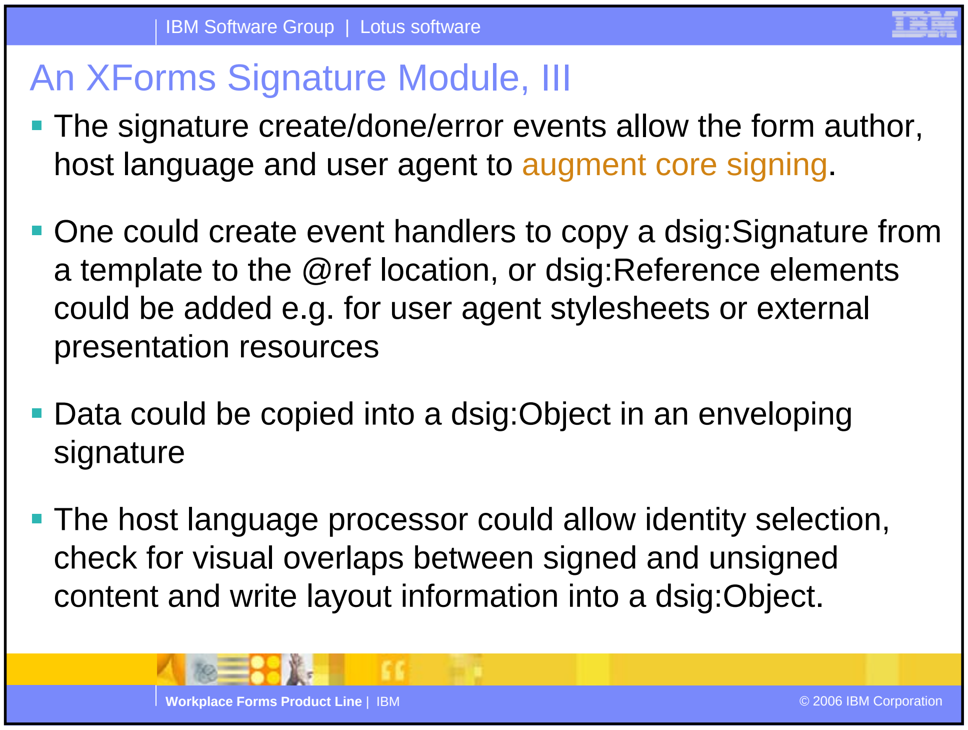
```
<submission id=“search”
    ref=“data to submit”
    action=“http://where to send data”
    instance=“where to put result data” ...>

    <action ev:event=“xforms-submit”>
       Actions to further prepare submission data
    </action>
    <action ev:event=“xforms-submit-done”>
       Actions to post-process submission results
    </action>
    <action ev:event=“xforms-submit-error”>
       Actions that respond to submission failure
    </action>
</submission>
```

An XForms Signature Module, I

- In XForms, changes to a form are made to instance data only, so the <dsig:Signature> must appear in an <xforms:instance>
- The <Reference> with no URI must take its initial resource from the originating document because instance data is maintained in separate DOMs from the originating document, so a same document reference indicates the XML data.
- An <xforms:signature> element should be added.
- A ref attribute would indicate the dsig:Signature to operate on, and child Actions would be allowed so that events targeted at the <xforms:signature> can be handled.
- Events should be created for signature creation/validation. Default processing should perform core signature operations.

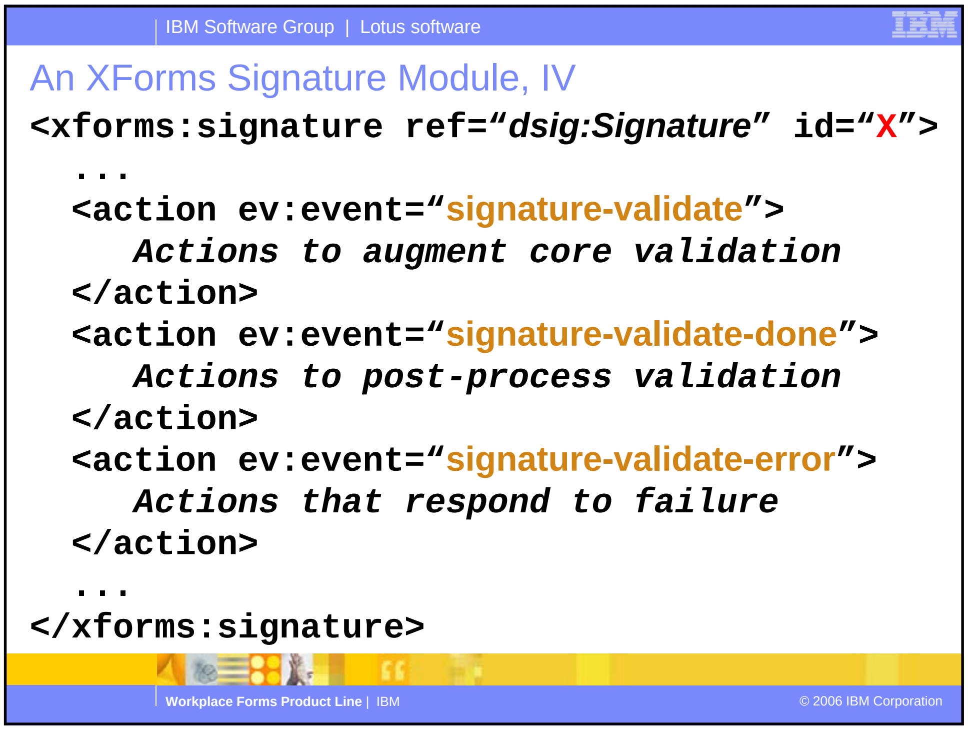
An XForms Signature Module, II

```
<xforms:signature ref="dsig:Signature" id="X">

  <action ev:event="signature-create">
     Actions to prepare dsig:Signature
  </action>
  <action ev:event="signature-create-done">
     Actions to post-process success
  </action>
  <action ev:event="signature-create-error">
     Actions that respond to failure
  </action>
  ...
</xforms:signature>
```



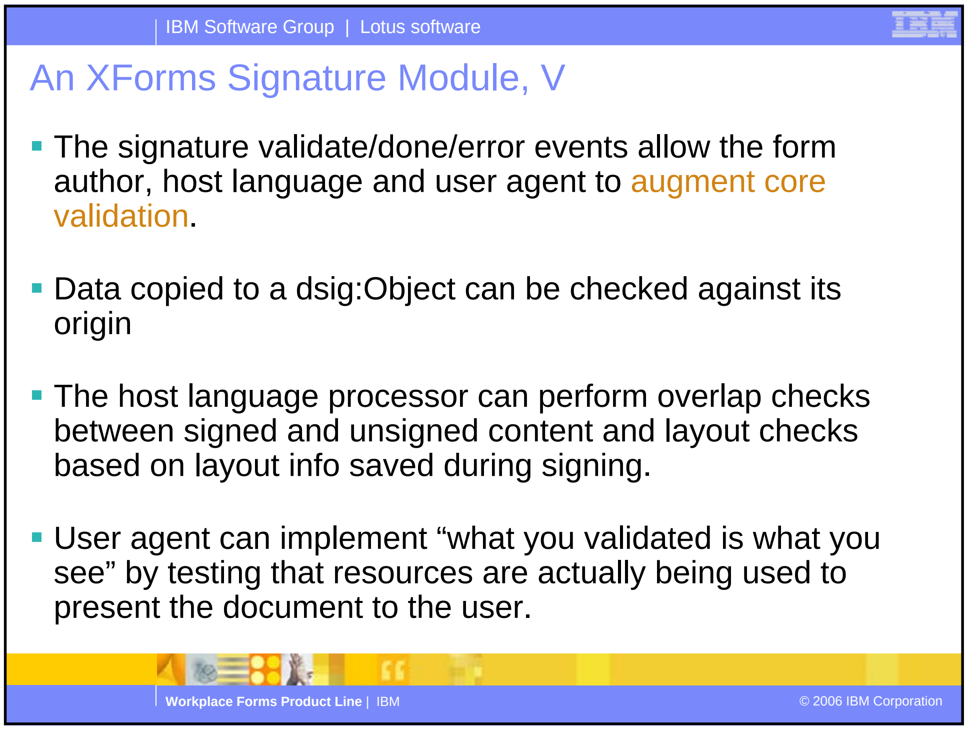
An XForms Signature Module, III

- The signature create/done/error events allow the form author, host language and user agent to augment core signing.
- One could create event handlers to copy a dsig:Signature from a template to the @ref location, or dsig:Reference elements could be added e.g. for user agent stylesheets or external presentation resources
- Data could be copied into a dsig:Object in an enveloping signature
- The host language processor could allow identity selection, check for visual overlaps between signed and unsigned content and write layout information into a dsig:Object.



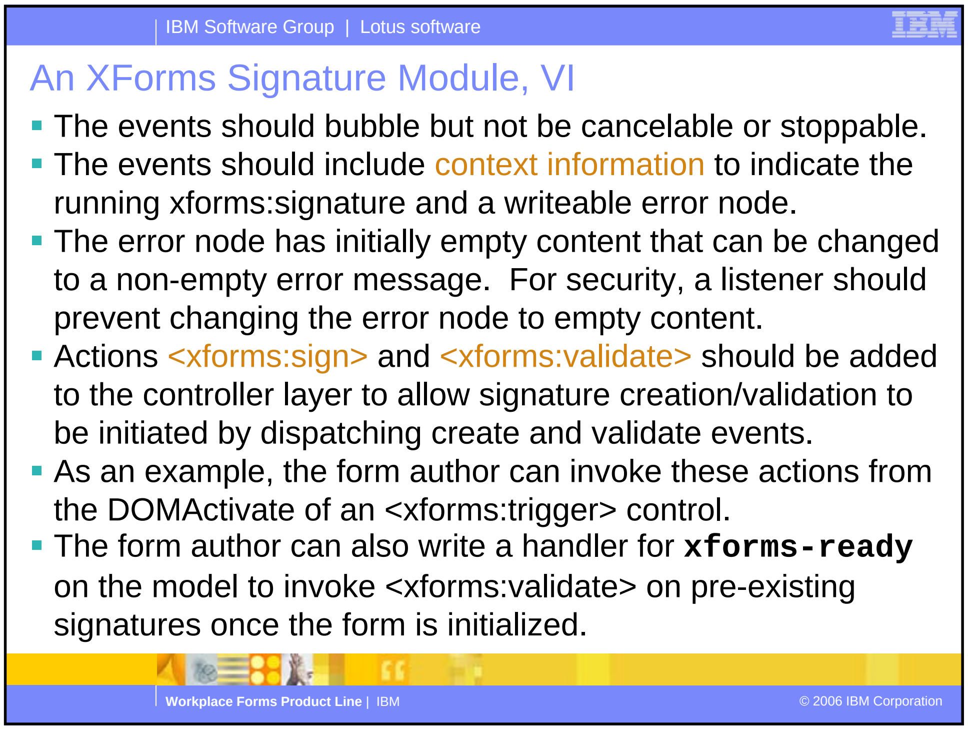
An XForms Signature Module, IV

```
<xforms:signature ref="dsig:Signature" id="X">
  ...
  <action ev:event="signature-validate">
     Actions to augment core validation
  </action>
  <action ev:event="signature-validate-done">
     Actions to post-process validation
  </action>
  <action ev:event="signature-validate-error">
     Actions that respond to failure
  </action>
  ...
</xforms:signature>
```



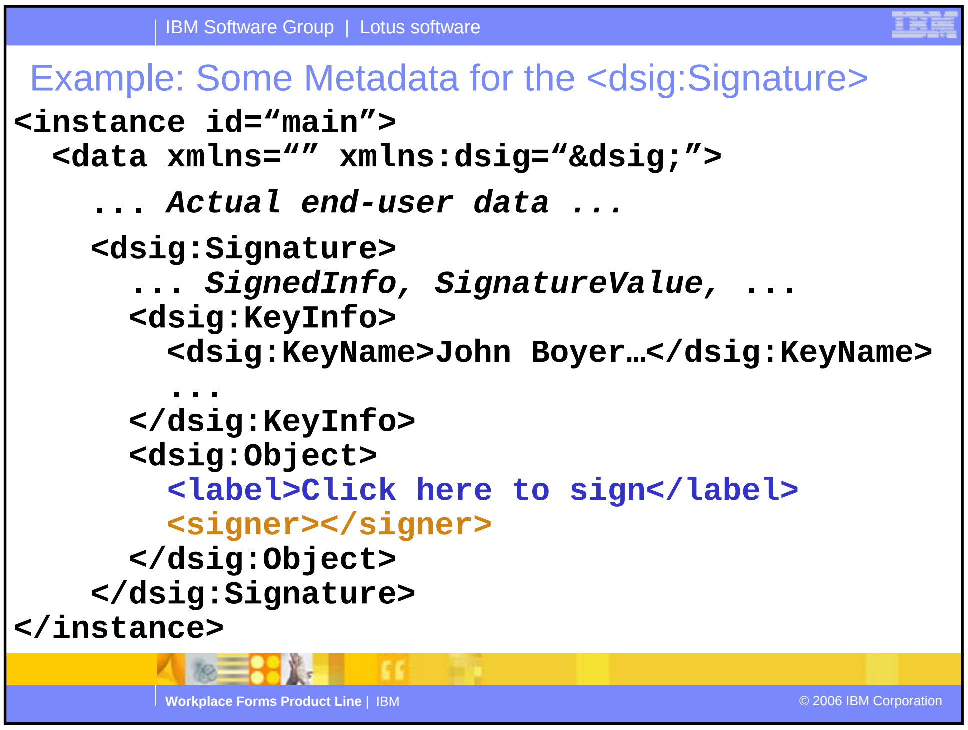
An XForms Signature Module, V

- The signature validate/done/error events allow the form author, host language and user agent to augment core validation.
- Data copied to a dsig:Object can be checked against its origin
- The host language processor can perform overlap checks between signed and unsigned content and layout checks based on layout info saved during signing.
- User agent can implement “what you validated is what you see” by testing that resources are actually being used to present the document to the user.



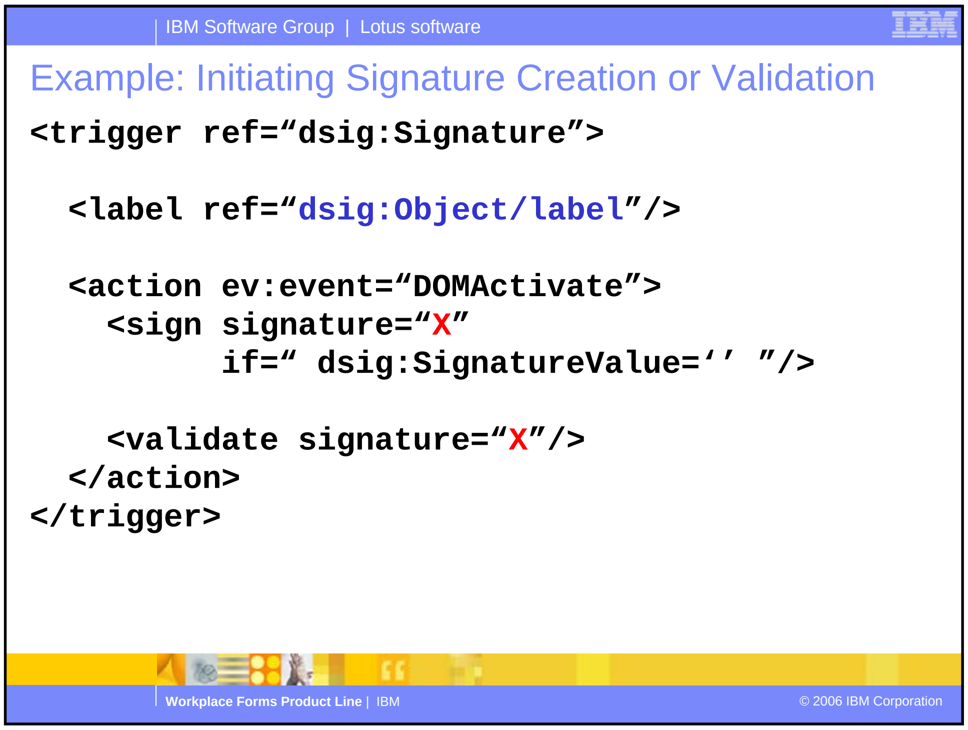
An XForms Signature Module, VI

- The events should bubble but not be cancelable or stoppable.
- The events should include context information to indicate the running xforms:signature and a writeable error node.
- The error node has initially empty content that can be changed to a non-empty error message. For security, a listener should prevent changing the error node to empty content.
- Actions <xforms:sign> and <xforms:validate> should be added to the controller layer to allow signature creation/validation to be initiated by dispatching create and validate events.
- As an example, the form author can invoke these actions from the DOMActivate of an <xforms:trigger> control.
- The form author can also write a handler for **`xforms-ready`** on the model to invoke <xforms:validate> on pre-existing signatures once the form is initialized.



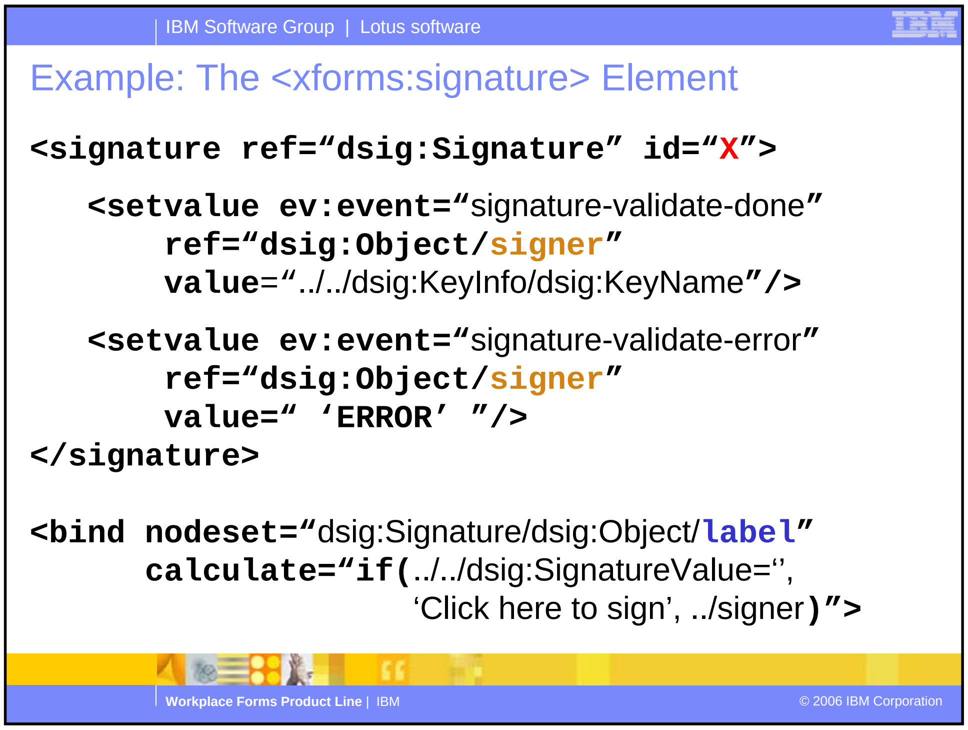
Example: Some Metadata for the <dsig:Signature>

```
<instance id="main">
  <data xmlns="" xmlns:dsig="&dsig;">
    ... Actual end-user data ...
    <dsig:Signature>
      ... SignedInfo, SignatureValue, ...
      <dsig:KeyInfo>
        <dsig:KeyName>John Boyer…</dsig:KeyName>
        ...
      </dsig:KeyInfo>
      <dsig:Object>
        <label>Click here to sign</label>
        <signer></signer>
      </dsig:Object>
    </dsig:Signature>
</instance>
```



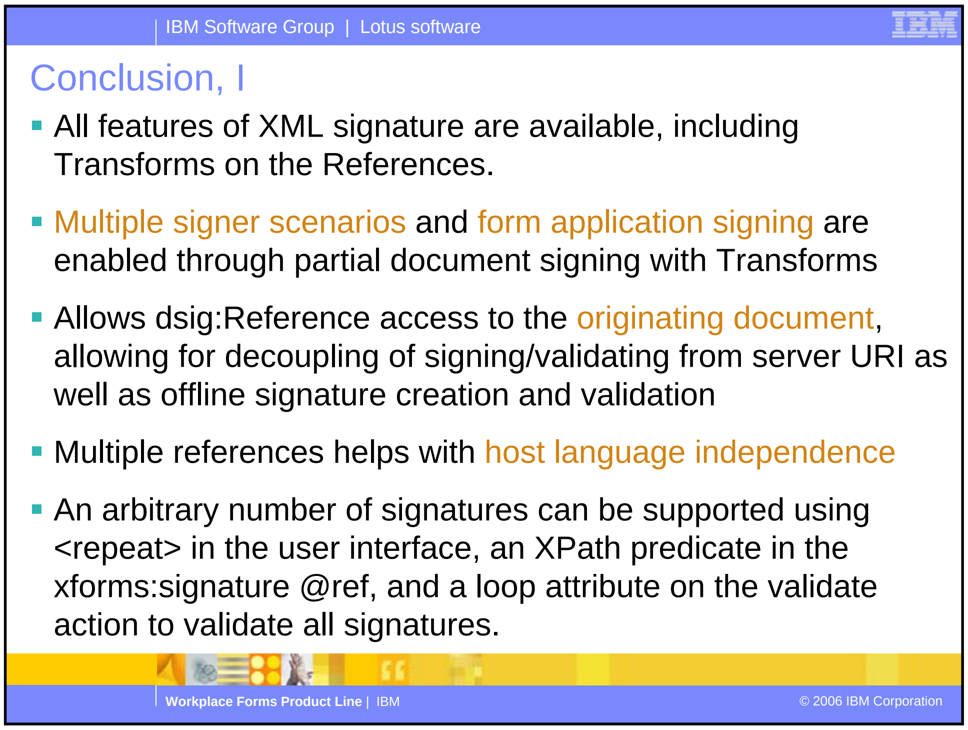
Example: Initiating Signature Creation or Validation

```
<trigger ref="dsig:Signature">

  <label ref="dsig:Object/label"/>

  <action ev:event="DOMActivate">
    <sign signature="X"
          if=" dsig:SignatureValue='' "/>

    <validate signature="X"/>
  </action>
</trigger>
```



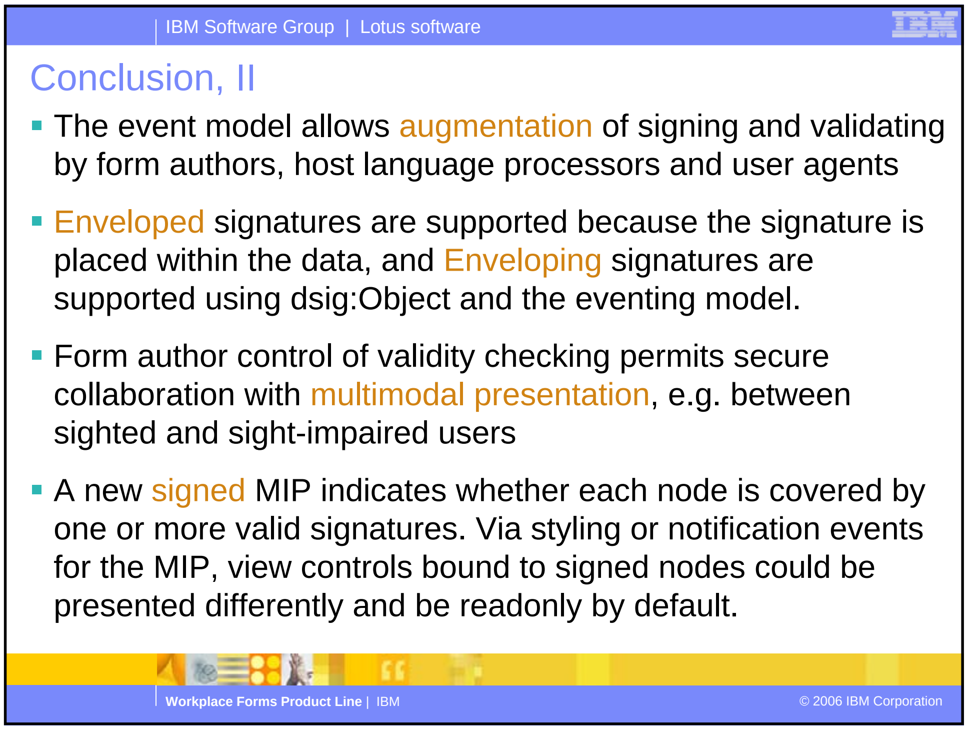
Example: The <xforms:signature> Element

```
<signature ref="dsig:Signature" id="X">

   <setvalue ev:event="signature-validate-done"
       ref="dsig:Object/signer"
       value="../../dsig:KeyInfo/dsig:KeyName"/>

   <setvalue ev:event="signature-validate-error"
       ref="dsig:Object/signer"
       value=" 'ERROR' "/>
</signature>

<bind nodeset="dsig:Signature/dsig:Object/label"
      calculate="if(../../dsig:SignatureValue='',
                          'Click here to sign', ../signer)">
```



Conclusion, I

- All features of XML signature are available, including Transforms on the References.
- Multiple signer scenarios and form application signing are enabled through partial document signing with Transforms
- Allows dsig:Reference access to the originating document, allowing for decoupling of signing/validating from server URI as well as offline signature creation and validation
- Multiple references helps with host language independence
- An arbitrary number of signatures can be supported using <repeat> in the user interface, an XPath predicate in the xforms:signature @ref, and a loop attribute on the validate action to validate all signatures.



Conclusion, II

- The event model allows augmentation of signing and validating by form authors, host language processors and user agents
- Enveloped signatures are supported because the signature is placed within the data, and Enveloping signatures are supported using dsig:Object and the eventing model.
- Form author control of validity checking permits secure collaboration with multimodal presentation, e.g. between sighted and sight-impaired users
- A new signed MIP indicates whether each node is covered by one or more valid signatures. Via styling or notification events for the MIP, view controls bound to signed nodes could be presented differently and be readonly by default.