

Design and Implementation of an XSL-T and XML-based Workflow System

Marc **Stauch** <msta@ovidius.com>

Robert **Tolksdorf**, Dr.-Ing. <tolk@cs.tu-berlin.de>

Abstract

Distribution of information and processes has become a prerogative of increased interest in workflow systems. With the new standards XML and XSL-T for processing information new paradigms for the architecture of such systems have emerged. The prototypical implementation of the "Workspaces" model that is to be presented in this paper relies heavily on XML and XSL-T for the encoding and generation of all of its defining and working data. The Intermediate data is generated from the XML sources using XSL-T.

Workflows in the Workspaces system are defined as XML documents that comply with a workflow definition DTD. This DTD has been derived from the Workflow Process Definition Language (WPDL) as proposed by the Workflow Management Coalition (WfMC) in its Process Definition Interchange Process Model (Interface 1). The individual running workflow instance is considered to be a sequence of document transforming steps that can be automatic or user-based. Steps are also used for coordination within the workflow instance. The generation as well as the coordination of the individual steps from this workflow definition are handled by XSL-T scripts. These scripts extract and merge all the required data from the different relevant XML sources which are potentially distributed.

The execution of the scripts and the processing of the results has been embedded in a Java-based Workspaces-engine that offers a hardware independent GUI. Given the decentralized nature of the Workspaces engines, the system not only offers possibilities for the distribution of information but also more interestingly for the distribution of processes.

1. Introduction

Integration of workflow management systems within Internet technologies has taken place to a certain limited extent during the last years. Distributed, ubiquitous documents placed in the world wide web representing the level of integration that has been achieved. However the traditional language of the net - HTML - will clearly be replaced in the near future by the much more powerful XML standard. Through the use of this standard the structure of information is regarded to be fundamentally independent of its presentation. Moreover with XSL-T a powerful standardized tool for the transformation of this type of information has been made available. Hence new possibilities of distributed information processing, especially for workflow management systems, emerge.

In the following we will present the architecture as well as the prototypical implementation of the Workspaces workflow model that was first proposed in [\[Tol00a\]](#) and [\[Tol00b\]](#). In the Workspaces system data of workflow instances is processed along the lines of these new paradigms, and moreover the definition of the workflows themselves is structured and processed according to the standards of XML and XSL-T. This paper will show how the use of the new standards supports heterogeneous, distributed organizations.

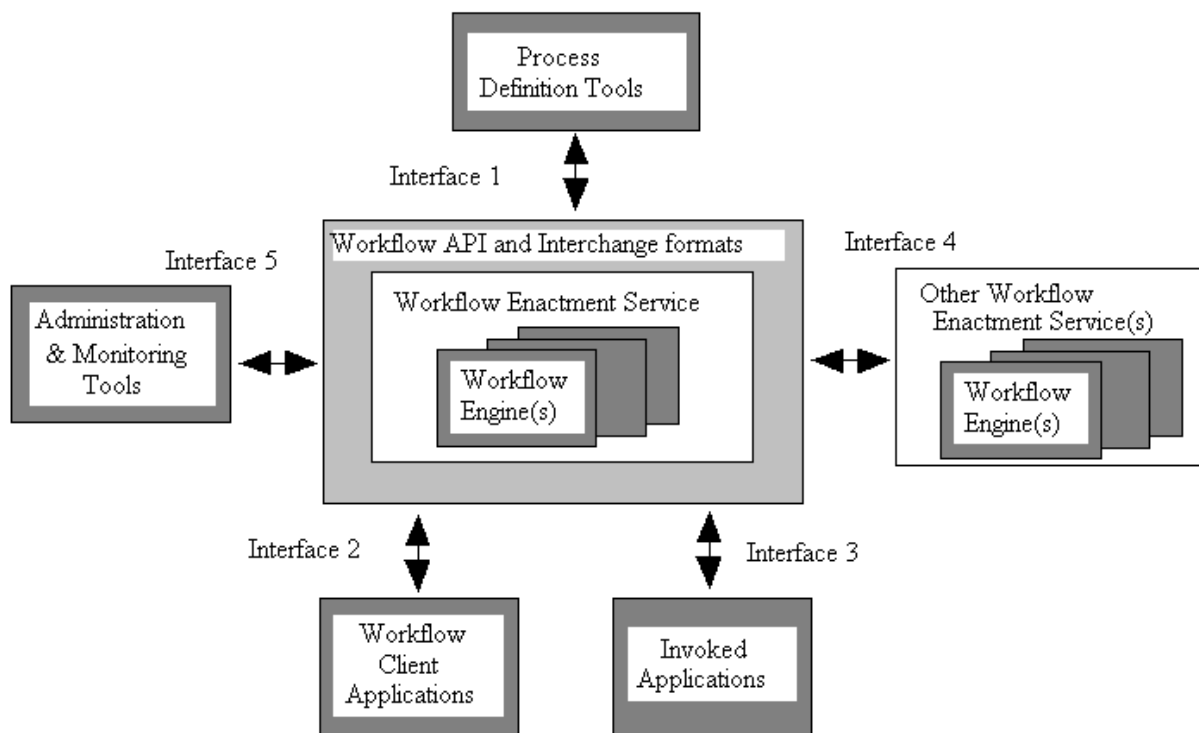


Figure 1. Definitions of the WfMC Reference Model

The industry consortium of the major workflow management system vendors, the Workflow Management Coalition (WfMC) has discussed the integration of workflow and Internet technologies ([Wor98b]). There has been a wide acceptance of the fact that these technologies are complementary. The main advantages are seen to be zero application deployment costs, ubiquity, integration tools, distribution and virtual enterprise, and electronic data interchange. The efforts in the WfMC aim primarily at the interface 4 of their reference model (Figure 1). The Workspaces model however introduces XML, XSL-T and Internet technology at the more basic level of interface 1, which is concerned with the data interchange between the definition of the workflow processes and the workflow enactment service (which we will refer to as the workflow engine).

Another important component of the Workspaces architecture is the underlying coordination technology that provides services to support the interplay of distributed

and asynchronous processes. The XMLSpaces Architecture which is to be embedded in the Workspaces implementation is a distributed repository for XML documents offering associative operations in the tradition of the coordination language Linda ([GC92]) to access distributed XML files. Since this paper will focus on the use of XML and XSL-T in the processing of working and defining data, the role of the coordination technology to be used is considered of lesser importance in the following and shall not be covered in full extent.

2. The Workspaces Model and Implementation

2.1. The Notion of Steps

As depicted in [Figure 2](#), the notion of a step involving an activity is at the core of the Workspaces system. A step is the transformation of a document into another document: In the case of Workspaces of an XML encoded document into an XML encoded document. This basic transformation is referred to as an activity. Wherever possible the Workspaces model involves an XSL-T processor with a corresponding XSL-T script to implement such activities.

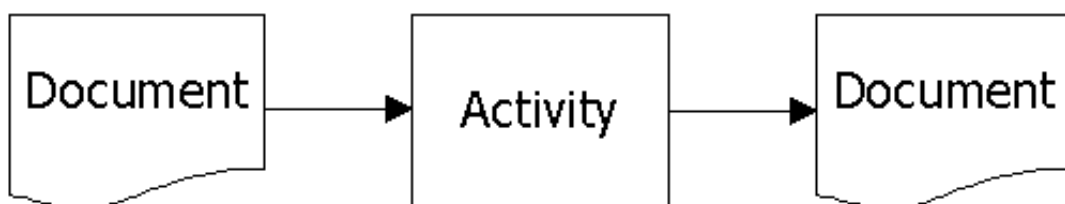


Figure 2. A simple Step in Workspaces

In Workspaces, we distinguish:

Basic steps

Basic steps are the basic activities and hence building blocks of workflows, they represent "normal" workflow steps and will be further subdivided into different

categories in the following.

Coordination steps Coordination steps are used to coordinate the flow of basic steps within workflow instances, refer to [Figure 3](#) for examples of split and join coordination steps. Note that these steps can be of a very high complexity involving logical constructs (or-join, and-join etc.) and conditions. As we will see in the following in the Workspaces model coordination steps are the only steps that cannot be implemented directly using XSL-T processors.

Meta steps Meta steps are higher level steps that operate on the definitions of workflows themselves. As we shall see in the following, structurally there is no difference between defining and working data in the Workspaces model, hence meta steps can be seen as a special kind of basic step in our model.

Furthermore basic steps can be subdivided into the following classes:

Automatic steps Automatic steps are responsible for compilation, transformation or formatting of documents by the workflow engine. These automatic transformations are implemented using XSL-T scripts in the Workspaces model.

External Steps External steps are the normal steps of an interactive kind, they involve the invocation of external applications with workflow documents. The user will process these documents and notify the Workspaces engine when the step has been completed. An example of this kind of step in the workflow instance depicted in [Figure 4](#) is the activity labeled "Answer" involving an external XML Editor.

User steps User steps represent unsupported work of the user. An example of this kind of step in the workflow instance in

figure [Figure 4](#) is the activity labeled "Think", the work here is to be done by the user without support (or document) from the workflow engine.

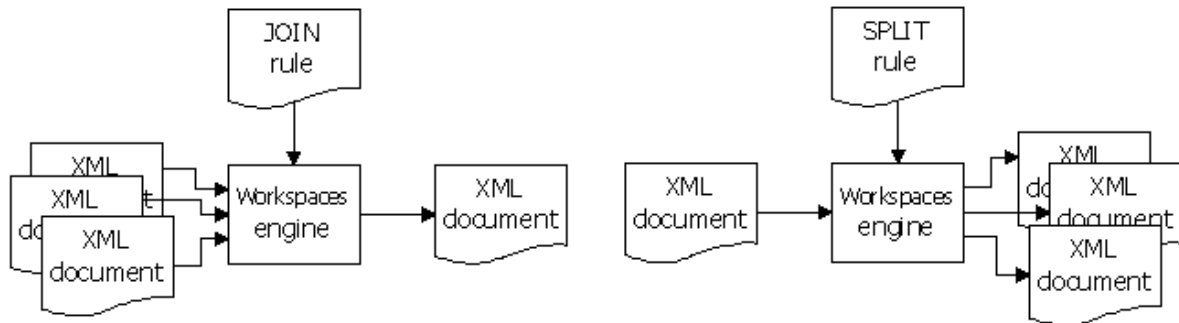


Figure 3. Join and Split Coordination steps

Refer to [Figure 4](#) for a simple example of linked activities within a workflow instance. This graph shows how coordination steps (here a "split" and a "join" step at the beginning and end of the workflow) and user as well as external steps ("Think" and "Answer") can be used to model the workflow of homework in a school-like environment.

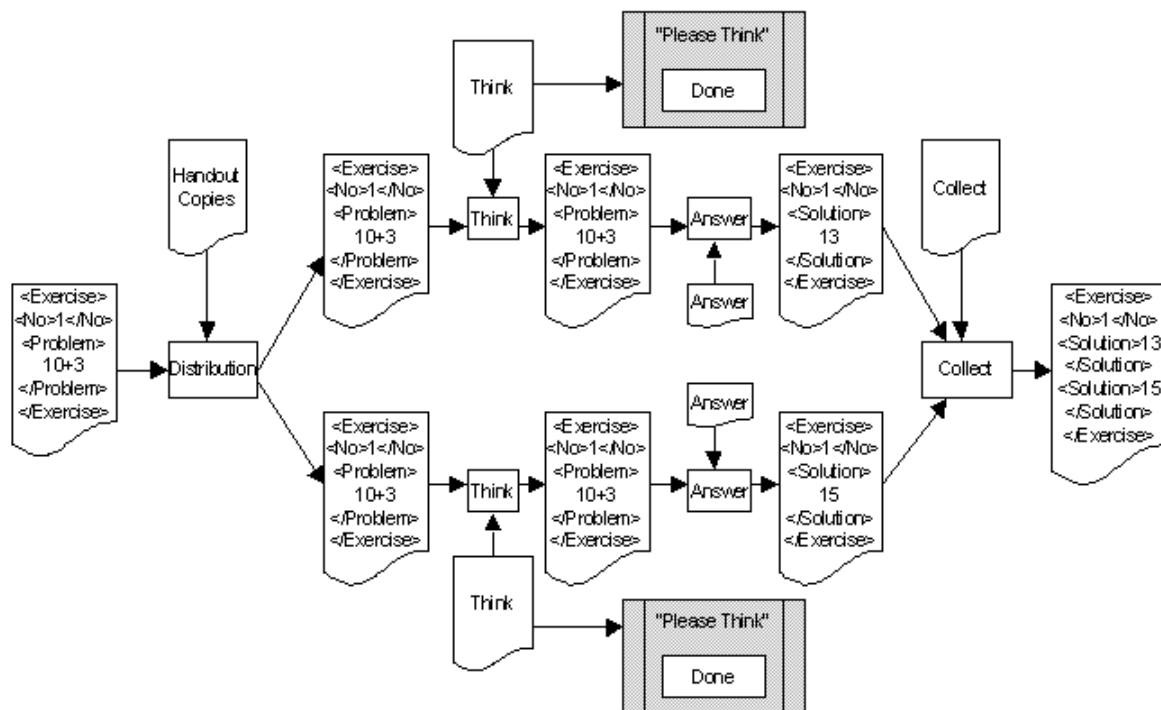


Figure 4. A complex Graph of Activities in a Workflow Instance

Through the Workflow Process Description Language (WPD) the WfMC offers a grammar for the description of such complex workflow definitions (in the interface 1 of its reference model). It represents the steps involved with all the necessary information needed to build a structural workflow definition in its grammar.

From the WPD the Workspaces Coordination Language (WSCL) DTD was derived for the Workspaces model. Hence a workflow definition in Workspaces is an XML document complying to the WSCL DTD.

2.2. The WSCL Document Type Definition

The basic structure and the dependencies between the building blocks of workflow definitions involved in the WPD are represented in figure Figure 5. The different elements in the corresponding Workspaces WSCL DTD have been derived more or less directly with some simple adaptations concerning mainly the typing of data. The complete code of the WSCL DTD is available at www.cs.tu-berlin.de/~stauch/Diplom/wkfl-dtd.

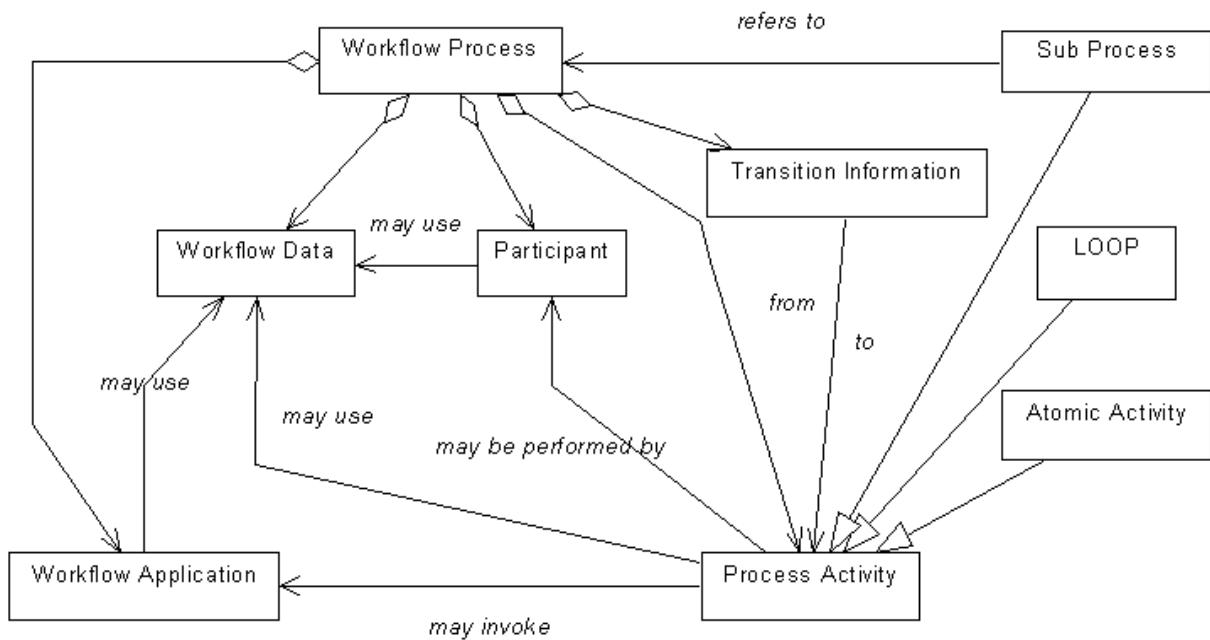


Figure 5. Elements of a Workflow Definition

The following XML code (a fragment in compliance with the WSCL DTD) in [Figure 6](#) describes a single activity of a workflow.

```

<ACTIVITY ID="1" NAME="Activity1">
  <APPLICATION PERFORMER="Henry" START-MODE="auto"
  INSTANCIATION="multiple"/>
  <ACCESSRESTRICITONS/>
  <TRANSITIONRESTRICITONS/>
  <DOCUMENT/></ACTIVITY>
  <TRANSITION ID="1" FROM="1" TO="2"/>
  <ACTIVITY ID="2" NAME="Activity2">
  <APPLICATIONS PERFORMER="John" START-MODE="auto"
  INSTANCIATION="single"/>
  <ACCESSRESTRICITONS/>
  <TRANSITIONRESTRICITONS/>
  <DOCUMENT/>
  </ACTIVITY>
  
```

Figure 6. Sample WSCL XML Code

2.3. Compilation of Steps

If the workflow definition can be considered to be a "program" written in a higher level language, then in the same vein the individual step in a workflow instance can be seen as the simpler "instruction" that the workflow engine will execute. Hence the main task of the workflow engine is the compilation of the process definition into smaller (possibly portable) step-instructions, in Workspaces this is performed by meta (XSL-T) steps as depicted in [Figure 7](#).

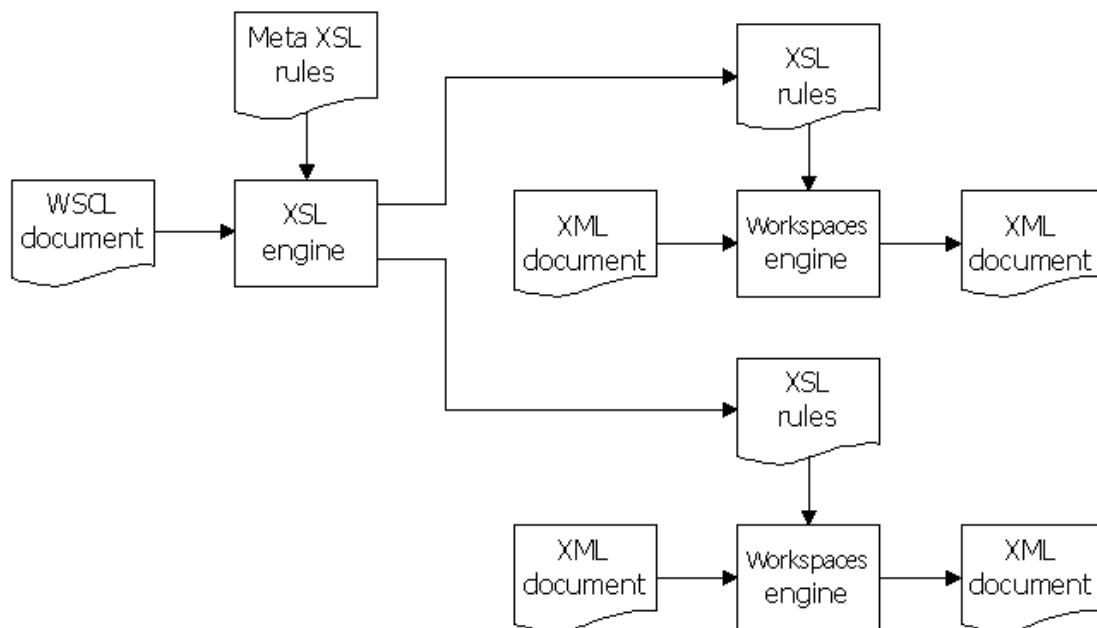


Figure 7. Compilation of Steps

In [Figure 7](#) we have referred to the step rules as XSL rules. However the rules will need to include additional data (not included in the current scope of XSL-T stylesheet possibilities) the reason for this being that the existing XSL-T engines are not powerful enough to execute all of the potential tasks of the Workspaces engine. It has been suggested that an existing XSL transformer will need to be extended to enable coordination (check for presence of documents, allow for multiple input files - which is presently not supported even though it is part of the W3 recommendation) and the execution of external programs. Currently coordination steps - being of a potentially highly complex nature, as well as external steps, launching user applications are implemented outside of the XSL-T processor.

As we have seen (refer to [Figure 6](#)) the WSCL encoded workflow definition consists at the most basic level of transitions between activities. The necessary information for the execution of a single step refers to exactly one activity and its transitions. The Workspaces meta-step that compiles the definition to extract information about a single step will thus need to find transitions that connect an activity with its corresponding successor activities.

Moreover the information needed to execute and coordinate the following activities needs to be extracted from the workflow definition. This involves:

- resolving variables within subflows with respect to their relative scope.
- extraction of a number of relevant elements from the XML workflow definition document that are referenced through their respective identifier attributes.

We refer to the resulting document as a Workspaces step document. This document is itself an XML document that can hence be validated.

Two alternative modes of compilation of the Workspaces definition into step documents are conceivable. We refer to these two alternative modes of compilation as the "eager" (run-time compilation) and "lazy" (start-time compilation).

Eager Compilation

This first compilation mode refers to compilation that will take place when or before the workflow is initialized. The necessary information for all the possible steps and variables of a workflow are generated when the workflow is launched. A repository will gather steps and variables. This repository could be implemented within XMLSpaces. Whenever a document refers to a certain following activity the corresponding step is extracted from XMLSpaces. Subflow definitions that are referenced within a workflow definition would have their own repositories and would hence not need to be compiled by the local engine.

Thus the repositories would allow for the sharing of step information across workflow instances. If externally

referenced subflow definitions would have many running workflow instances a repository would be interesting to allow code-sharing. Moreover updates of subflows would be left to their providers, ensuring soundness and completeness across large networks. New versions of steps would automatically be used by the workflow instances participants.

In the field of detached computing the notion of a step repository is of special interest. Relevant steps for a certain workflow instance could easily be packaged and transferred to the detached system along with pointers to documents within XMLSpaces. Working offline would become possible in such a way.

Lazy Compilation

This scenario refers to compilation that takes place as late as possible. Step information for a running workflow instance is only generated when is needed. Hence compilation takes place whenever a single activity has been completed.

In this mode of compilation there would be no unnecessary space consumption by potentially unused steps as would potentially be the case in the previous scenario. References to subflows and variables are resolved at the latest possible time thus updated variables and subflow definitions will automatically be used whenever available. Moreover a provider of subflow definitions is not required to compile his scripts. The risk of large scale server-side computation is reduced.

Since the step information is available at the same time as the updated current workflow document, this last document (extracted from XMLSpaces) could be included in the step XML document. In this way all the relevant information (including the document to be transformed) is

gathered within a single XML file.

Client-side lazy compilation of steps in a mobile environment would be more difficult. However copying parts of workflow definitions as well as relevant documents from XMLSpaces onto detached systems could still make offline compilation possible. Algorithms for determining and computing potential steps for future detached work within a workflow instance are conceivable.

The current prototype of the Workspaces engine uses the lazy mode of compilation. Ensuring consistency and soundness while still offering a high degree of distribution was our primary objective. We did not want to emphasize server-side computation as in traditional client-server architectures, which would clearly be a consequence of a step repository. Ubiquitous engines depending on a ubiquitous XMLSpaces implementation were favored in our system.

```
<!-- Insertion of Performers -->
<xsl:template mode="insert-performer" match="*">
  <xsl:variable name="name" select="@PERFORMER"/>
  <xsl:choose>
    <!-- is the matching PARTICIPANT in the WORKFLOW ? -->
    <xsl:when test='ancestor::WORKFLOW/child::PARTICIPANT[@ID=
name][position()=1]"/>
  </xsl:when> <!-- is the matching PARTICIPANT in the MODEL ?
-->
  <xsl:when
test='/child::MODEL/child::PARTICIPANT[@ID=name][position()=1]"/>
  </xsl:when> <!-- where is the PARTICIPANT ? -->
  <xsl:otherwise>
    <xsl:text>NO MATCHING PARTICIPANT FOUND</xsl:text>
  </xsl:otherwise> </xsl:choose>
</xsl:template>
```

Figure 8. XSL-T Template for the Insertion of Performers

Refer to [Figure 8](#) for an example of an XSL-T stylesheet template used in the Workspaces engine. This very simple template illustrates the insertion of the

responsible performer once an activity for a step has been determined.

Further information on the implementation of the Workspaces model is available at <http://www.cs.tu-berlin.de/~stauch/Diplom>.

2.4. Coordination: XMLSpaces

Since the execution of steps, as described above, is dependent on the presence of a document to be processed, the coordination of the flow is data driven, as opposed to control-driven. As has been pointed out in the introduction, Workspaces uses XMLSpaces - a Linda derivative - to coordinate its flows. XMLSpaces relies on its documents being XML encoded and offers a number of XML specific retrieval and query operations.

Workspaces is implemented by a number of identical engines that are located on different hardware platforms on different locations. The engines are coordinated through the presence or absence of special (user-dependant) documents. The resulting architecture is depicted in [Figure 9](#).

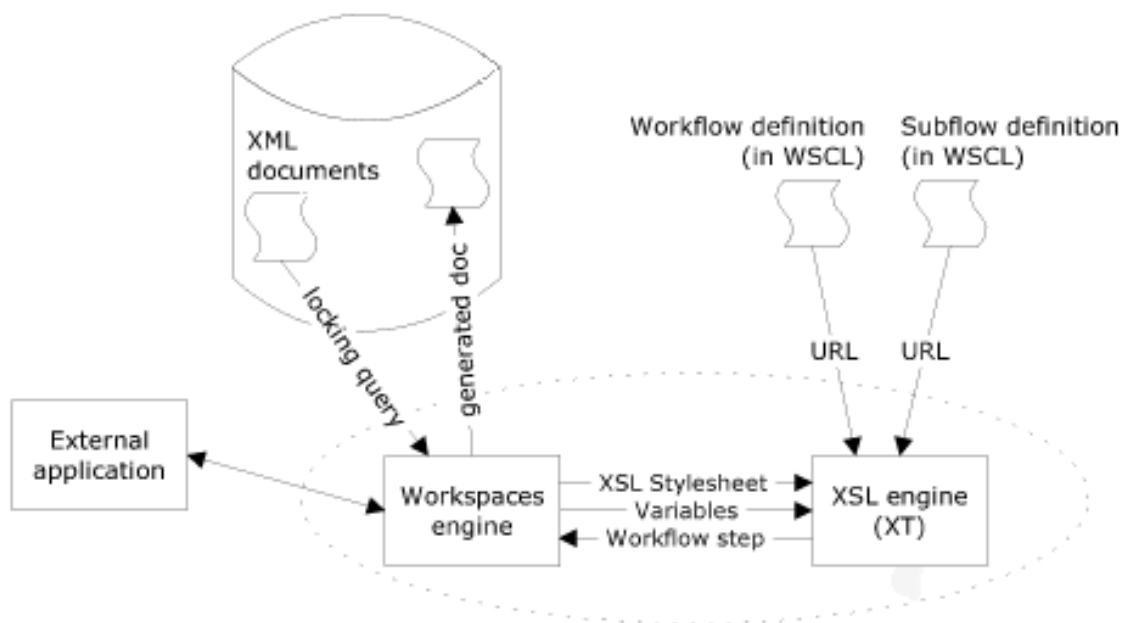


Figure 9. Workspaces Architecture

3. Implementation

The current prototypical implementation ([Sta99]) is based on a engine that launches external applications and coordinates the processes. It uses James Clark's Java implementation of XT as an XSL-T processor. Through XT the workflow definitions are compiled and the resulting individual steps that include all the necessary information for a single activity are then executed by the Java engine. The engine, through a simple Java Swing graphical user interface presents the available worklist to the user, who will choose which steps are to be executed (refer to figure [Figure 10](#)).

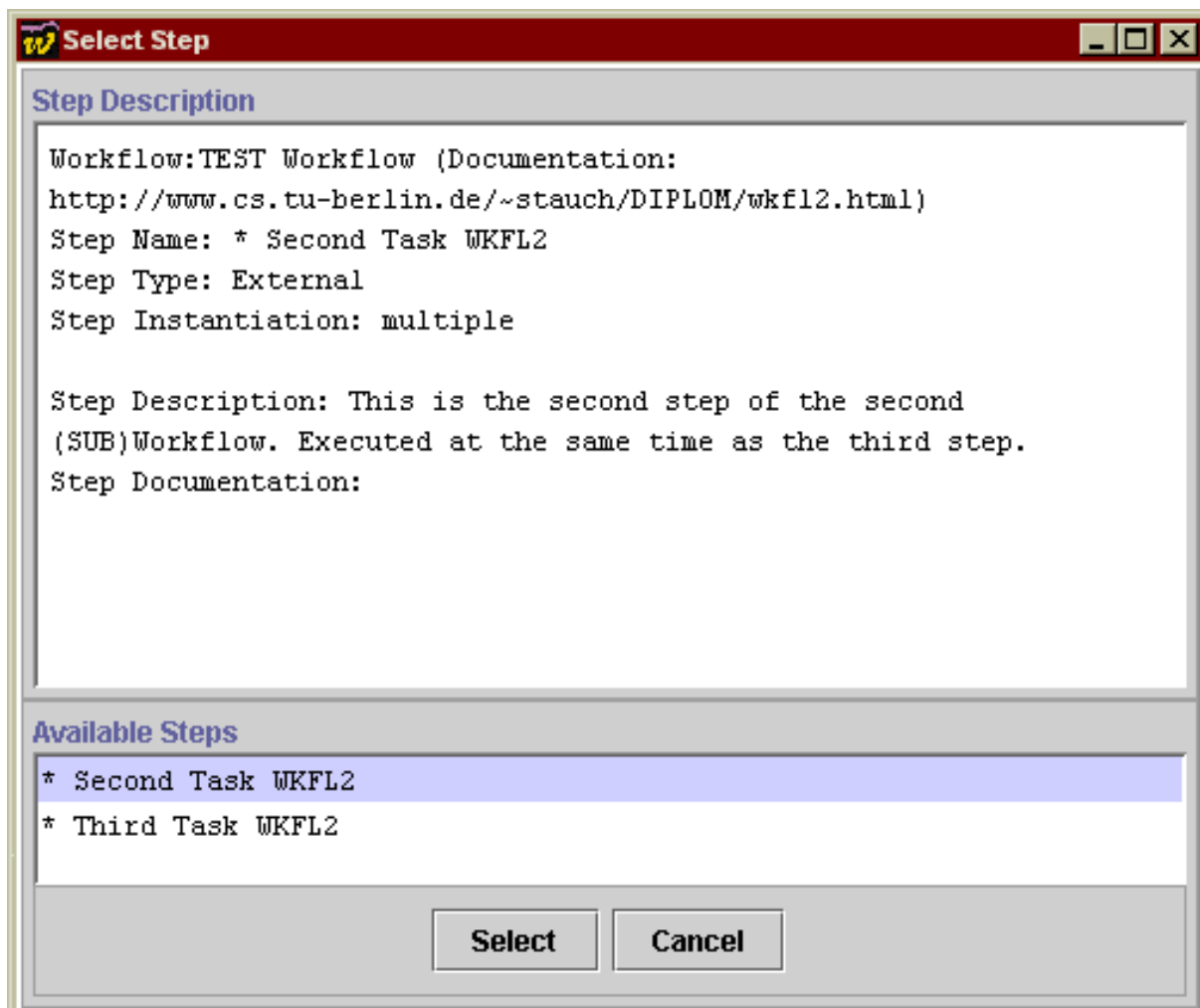


Figure 10. Screenshot of the Workspaces Prototype

4. Conclusions

In a restricted test environment first tests with the prototypical implementation have shown good performance and stability. The problems faced in real world applications may however be of a different nature, scalability is clearly an issue considering the memory and performance problems inherent in the DOM Model and the XSL-T transformation tools. The architecture of Workspaces however allows for integration of new transformation tools, should these problems be tackled and resolved in the future, it would be easy to adapt the engine to using new transformation engines.

In the field of detached computing these issues seem to be of relevance, since memory as well as computational power are very limited in these environments. New compilers with enhanced performance would allow for potentially small and portable Workspaces engines.

In the same way client-side browser support for XSL-T would open attractive possibilities for our architecture. A separation of the Workspaces engine from the XSL-T processor will then become possible.

The use of document based technologies for the modelling of processes has proved to be a success. However the XSL-T scripts involved have tended to be complex and clumsy. This might be a question of taste.

Our experiences have shown that building on standard technologies is the right way to go. We benefited greatly from the existence of tools and libraries to process standard formats. While this is an implementation advantage, we also believe that the resulting system is of higher value by supporting standard formats and technologies. Thus, we believe that XML and XSL-T can in fact serve as a basis for the implementation of a complex application such as a Web-based workflow system. Further information about Workspaces is available at <http://www.cs.tu-berlin.de/~tolk/workspaces>.

Bibliography

[CTV+98] Paolo Ciancarini, Robert Tolksdorf, Fabio Vitali, Davide Rossi, and Andreas Knoche. Coordinating Multiagent Applications on the WWW: A Reference Architecture. *IEEE Transactions on Software Engineering*, 24(5): 362-375, May1998.

[MC94] T. W. Malone and K. Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1): 87-119, 1994.

[GC92] David Gelernter and Nicholas Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2): 97-107, 1992.

[Sta99] Marc Stauch. Design and Implementation of a System for Distributed Workflows using XML / XSL. Master's Thesis, Technische Universität Berlin, 1999. <http://www.cs.tu-berlin.de/~stauch/diplom>

[TS01] Robert Tolksdorf and Marc Stauch. Using XSL to Coordinate Workflows. In *Kommunikation in Verteilten Systemen (KiVS)*, pp 127-138, Springer 2001

[Tol00b] Robert Tolksdorf. Coordination Technology for Workflows on the Web: Workspaces. In *Proceedings of the 4. International Conference on Coordination Models and Languages COORDINATION 2000*, LNCS 1906:36-50. Springer 2000.

[Tol00a] Robert Tolksdorf. Coordinating Work on the Web with Workspaces. In *Proceedings of the IEEE Ninth International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises WET ICE 2000*. IEEE Computer Society Press, pp 248-253, 2000.

[Wor98a] Workflow Management Coalition. Interface 1: Process Definition Interchange Process Model, 1998. <http://www.wfmc.org>

[Wor98b] Workflow Management Coalition. Workflow and Internet: Catalysts for Radical Change. WfMC White Paper, 1998. <http://www.wfmc.org>

Glossary

WfMC	Workflow Management Coalition
WPDL	Workflow Process Description Language
WSCL	Workspaces Coordination Language

Biography

Marc Stauch

Systems Engineer
OVIDIUS GmbH
Berlin
Germany
Email: msta@ovidius.com

Marc Stauch - Marc Stauch attended the Australian National University, Free University Berlin, and the Technical University Berlin. He holds a Master of Computer Science from the Technical University Berlin. He has extended experience in the fields of SGML and XML technologies.

Marc Stauch is currently working for OVIDIUS as a consultant and systems engineer.

Robert Tolksdorf, Dr.-Ing.

Assistant Professor
Technische Universität Berlin
Computer Science, FR 6-10
Franklinstr. 28/29
Berlin
Germany
Email: tolk@cs.tu-berlin.de

Dr.-Ing. Robert Tolksdorf - Dr.-Ing. Robert Tolksdorf is Assistant Professor at the study group Formal models, Logic and Programming at the Technical University of Berlin, Department of Computer Science. He received his Dr.-Ing. in computer science from the Technical University Berlin in 1995. His main research interests include foremost coordination languages and models in the context of open distributed systems and Web-technologies and application areas like workflow management. A second focus of his work is on document markup languages in the Web context.

He is the author of several journal articles, four books on Internet and coordination technology and several refereed publications, invited book

contributions and research reports. He is series editor of xml.bibliothek, a book series on XML based technologies with dpunkt.verlag, Heidelberg.

He served in various program committies and is currently member of the Steering Committees of the International Conference on Coordination Languages and Models, Coordination and of the IEEE WETICE conference series. He is member of the steering committee of the GI Fachgruppe 4.9.2 on Multimedia which includes XML-technologies.