Standardizing Software Documentation Using XML

John E. Cooke - University of Saskatchewan

- Jim. E. Greer - University of Saskatchewan
- Judi Thomson - Pacific Northwest National Lab
- Jeff Dyck - Hamilton Myriadgate Technology
- Reagan Penner - PMC-Sierra

Abstract

Firms that develop software must document both the software product and the process by which it was developed. Information from past projects is invaluable for improving the software production process. Information such as time-lines, critical paths, estimations and process metrics is often kept, but seldom kept in a standardized manner that permits retrieval. XML provides a means of implementing a standard set of documents in a manner that permits retrieval across many different projects. A basic set of documents will be defined, and work in implementing this document set in XML for a small software firm will be described.

Background – Basic Concepts

- A software project.
- Software documentation.
- Components of software documentation.
- An information base for software documentation.
- Retrieval from that information base.

A Software Project

- Some specific piece of software as the main deliverable.
- Software made by a specific group of developers for a defined group of users.
- Made within constraints of time, money, and quality.
- Made by following a particular defined process.

Software Documentation

- Facts about the software itself (the product):
  - for using the current version of the software,
  - for developing/enhancing/fixing the software.
- Facts about the software development (the project)
- Production of software documentation:
  - as a by-product of the development process,
  - as a separate activity by professional writers.

Components of Software Documentation

- Text.
- Pictures and diagrams.
- Code or prototype examples.
- Tables, lists and numbers (static media).
- Video and audio (dynamic media).

An Information Base for Software Documentation

- Organized and complete repository of software documentation artifacts.
- Should these artifacts (or their metadata) be stored as XML?
- What must be included in the information base?
- How can it be implemented?

Retrieval from a Software Information Base

- Retrieval by specification of various components.
- Retrieval based on a query.
- Retrieval based on defined needs.
- Retrieval based on assumed needs (adaptation).
- Presentation in various forms (more adaptation).

Background – Users and Developers

- Users of software documentation.
- Purposes in using software documentation.
- Problems in using conventional software document.
- Problems in developing software documentation.

Users of Software Documentation

- Developers.
- Users of the software.
- Maintainers.
- Managers.
- Auditors.

Purposes in Using Software Documentation

- To answer specific questions.
- To obtain general understanding.
- To learn how to use (help and tutorials).
- To review performance.
- To identify causes of observed problems.
- As the start to a redesign process.

Problems in Using Software Documentation

- Multiple formats and platforms (no integration).
- Incomplete or unavailable.
- Not current, inaccurate.
- Access methods don’t match needs (i.e. a pdf document rather than a tutorial).
- Mismatch between level of documentation and skills of user.

Problems in Developing Software Documentation

- Multiple formats and platforms.
- Incompatible skills (person and application, person and knowledge).
- Documentation is often an extra task with no immediate reward or necessity.
- Need for access control and configuration management (of documentation as well as code).

Big Question Number One

If software documentation is a problem that has been solved, why are so many “how to” books sold for any common software product, and why is a great deal of training required for most in-house software systems?

Big Question Number Two

Software documentation and software project documentation have traditionally been treated as two sets of distinctive problems. Is it reasonable to think of software documentation as a single problem with many interrelated parts?

Big Question Number Three

If software documentation can be treated as a (massive) single problem, does XML and its related technologies offer the potential for an integrated solution?

Three Research Projects Involving Software Documentation and XML

- Reagan's work (using an XML framework to link software document management and source code management systems).
- Jeff's work (developing new forms of XML-based software documentation and investigating how to develop and how to use it).
- Judi's work (semi-automated generation of tutorials from objects represented in XML).

Reagan's Project
XML-ISDE

An XML framework for integrating document management and source code management systems.

The Firm in Question

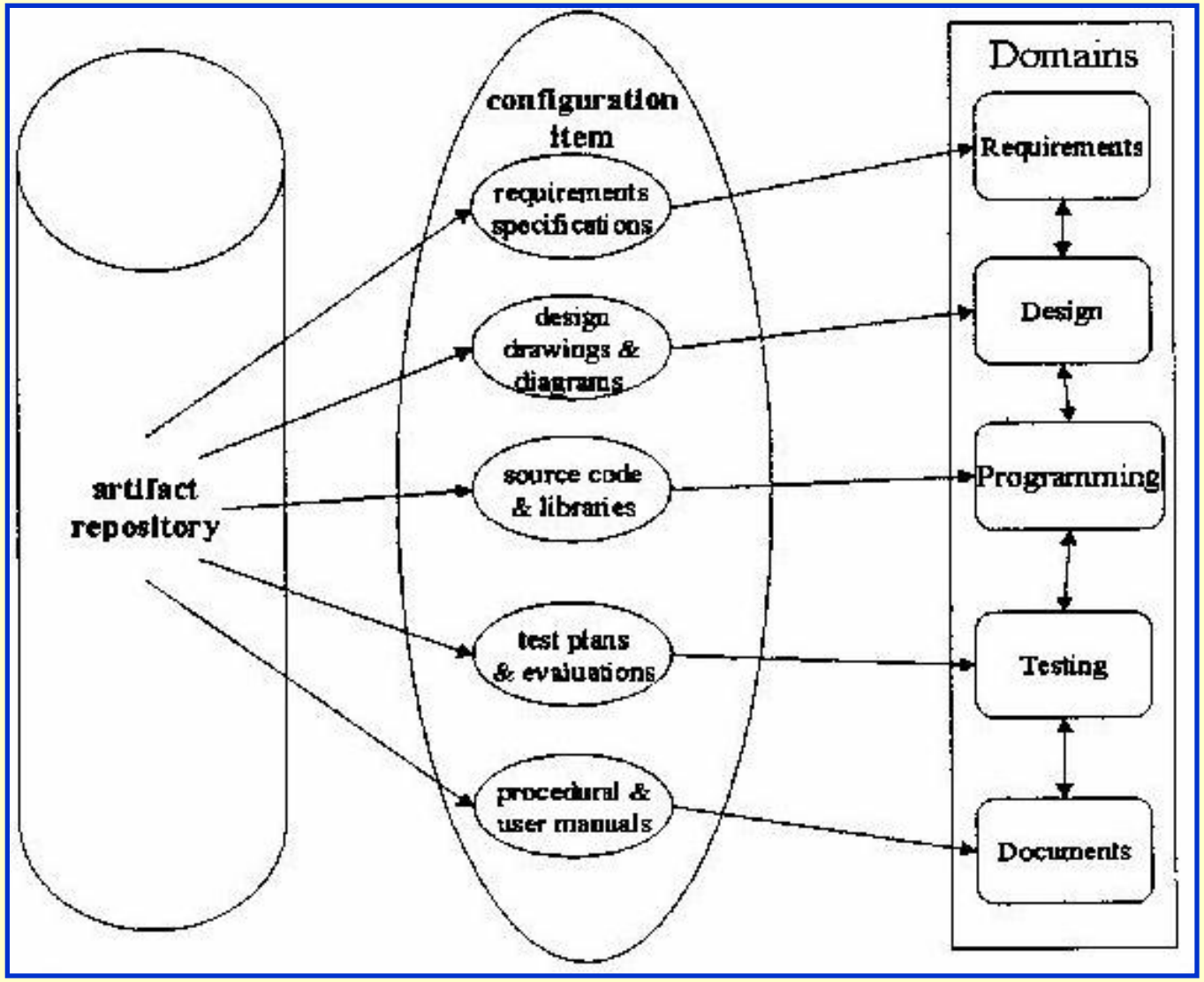
- Manufacturer of devices with embedded software.
- Strong emphasis on configuration management for both device and software.
- Completely separate systems for source code control and documentation.
- No major change considered possible.

Research Goals

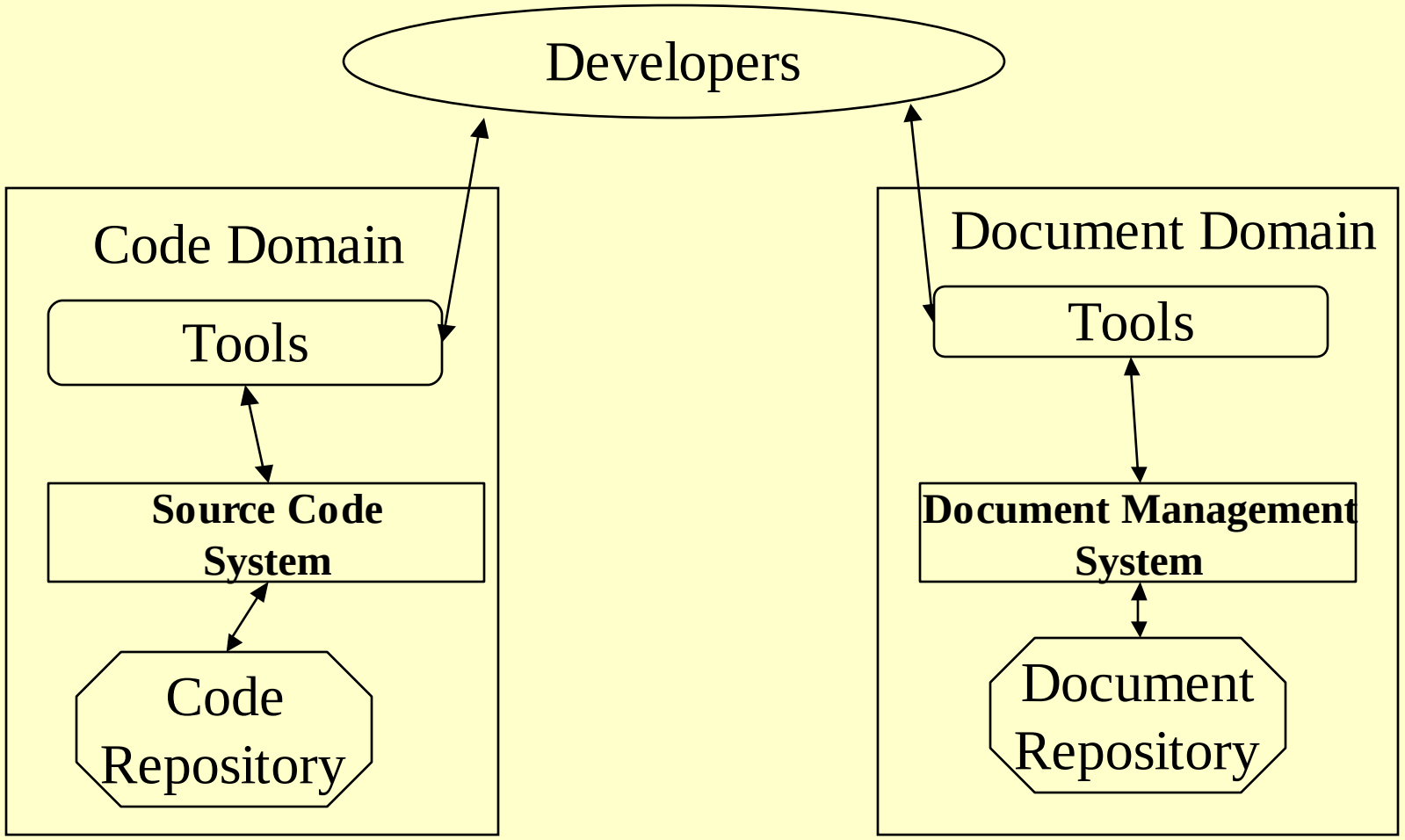
To solve problems found in current software development environment (SDE).

- Promote an integrated SDE than can be browsed.
- Link source code with its associated documentation.
- Support versioning in the integrated SDE.
- Ensure flexibility for future change.

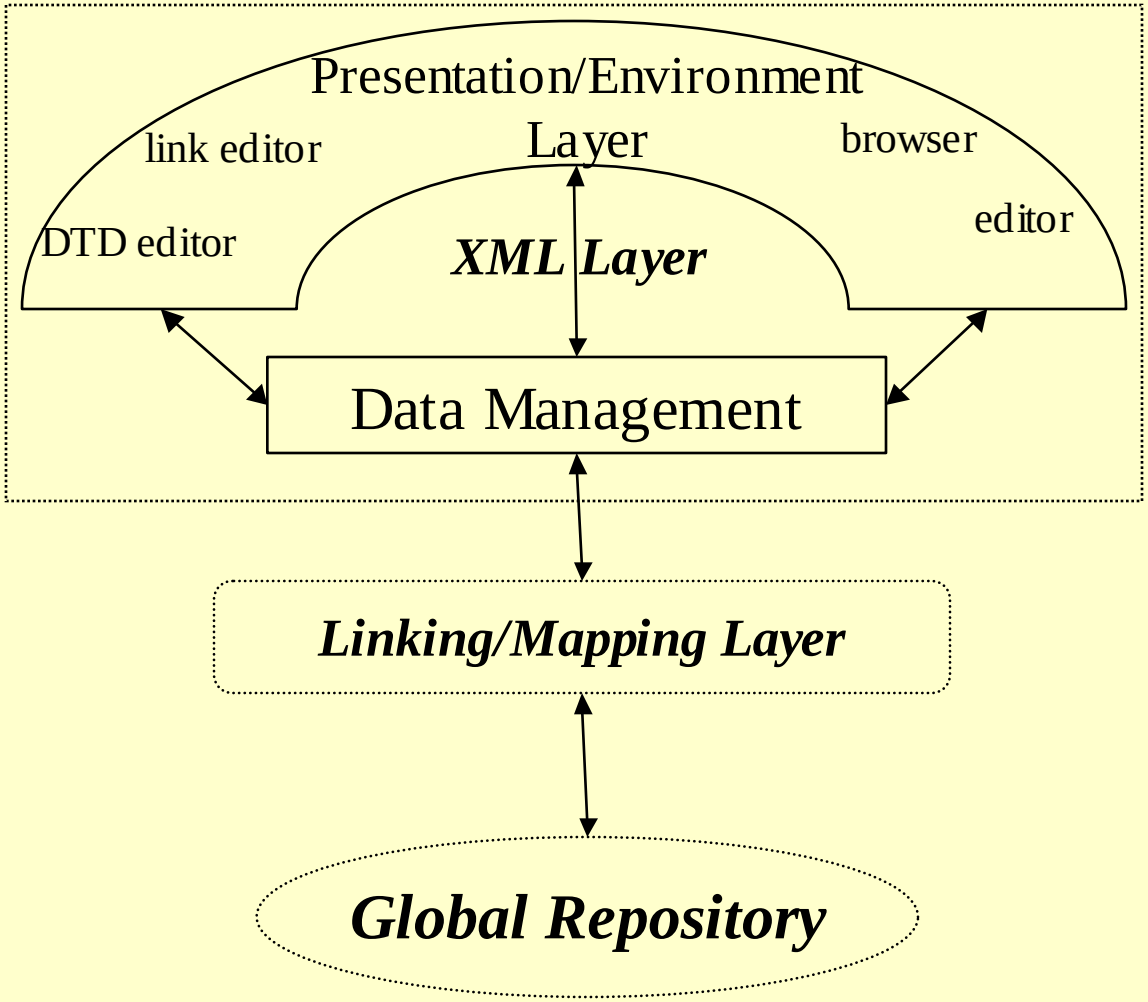
Integration Model



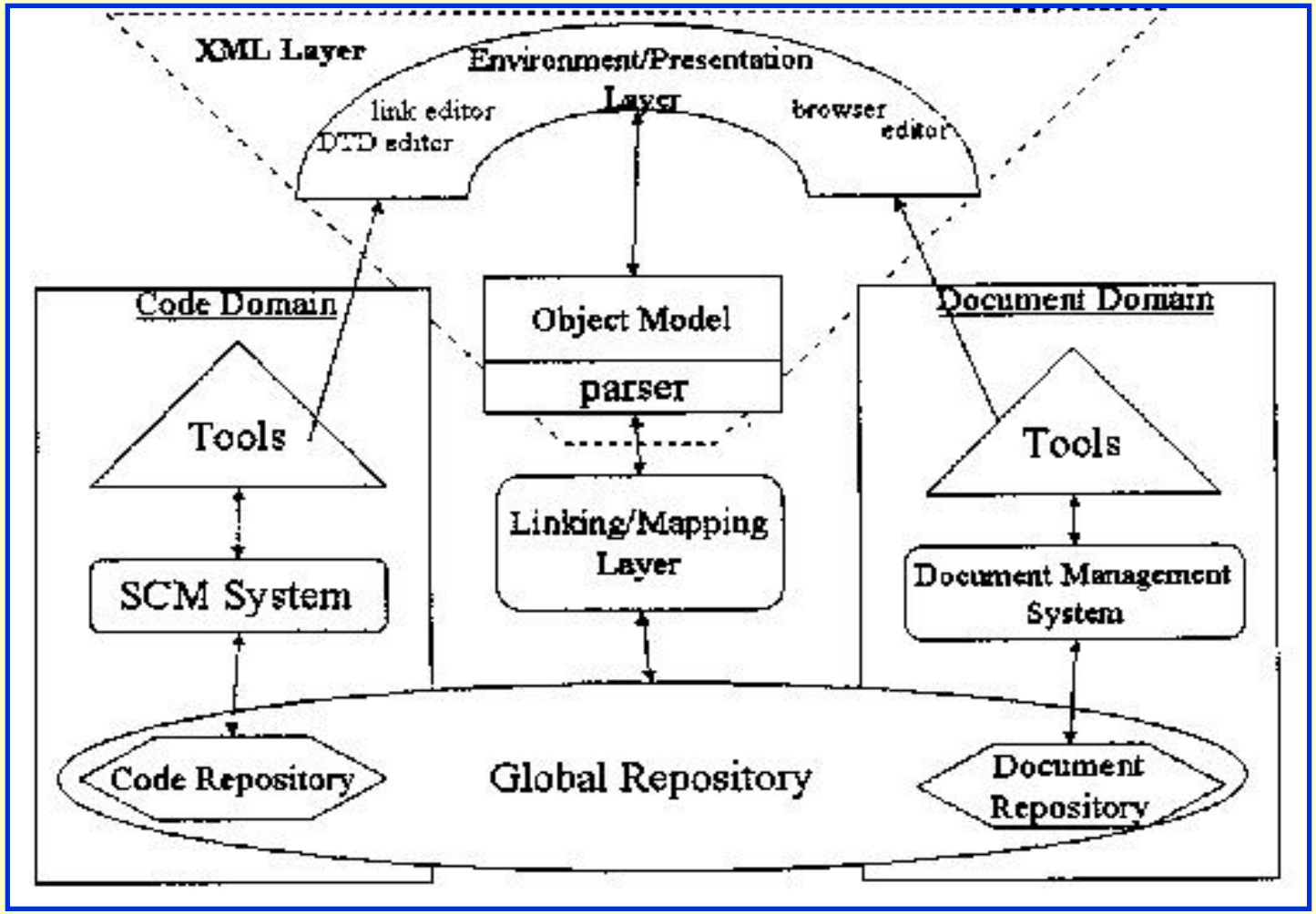
The Existing SDE



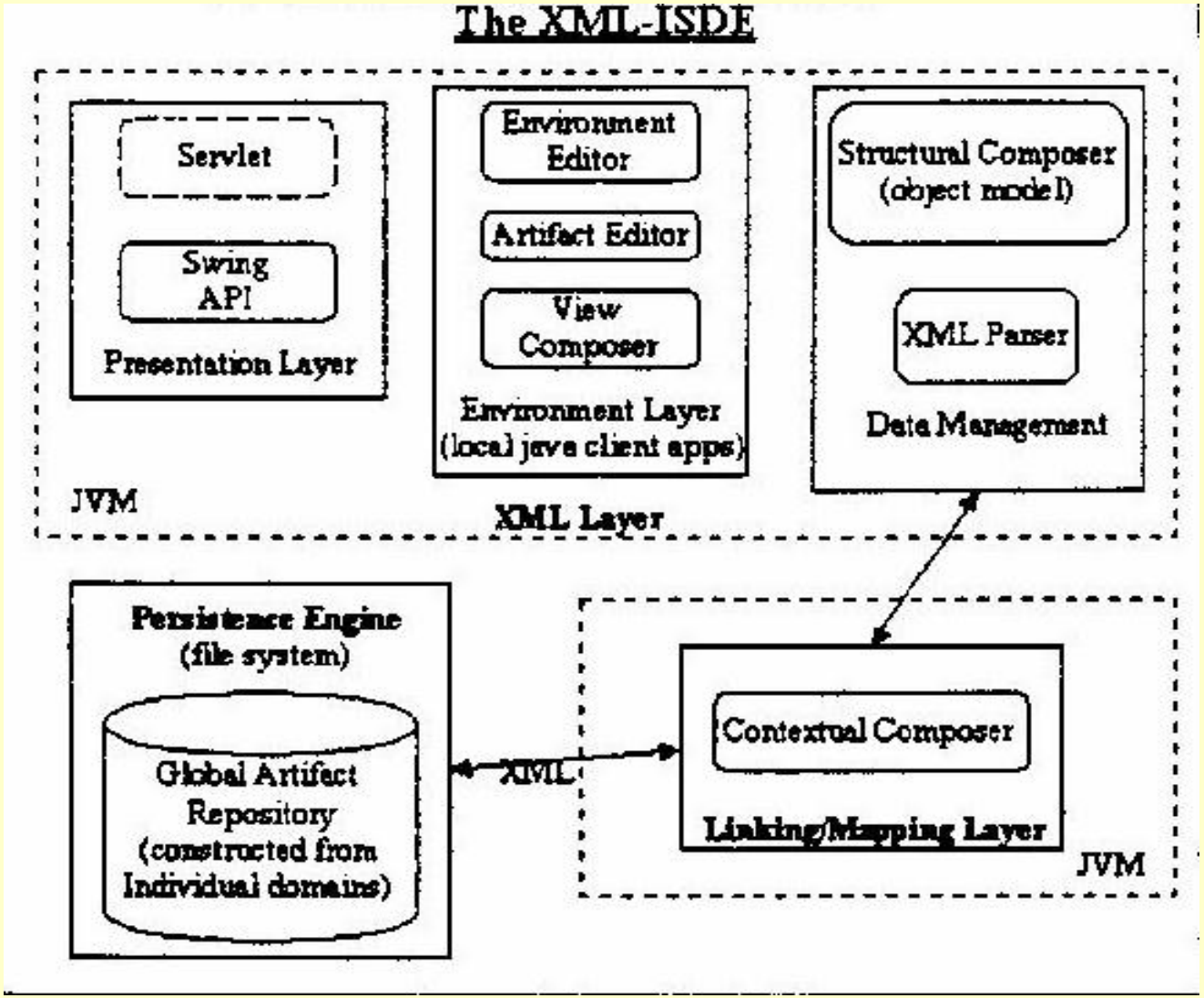
The XML-ISDE Framework



Domain Integration



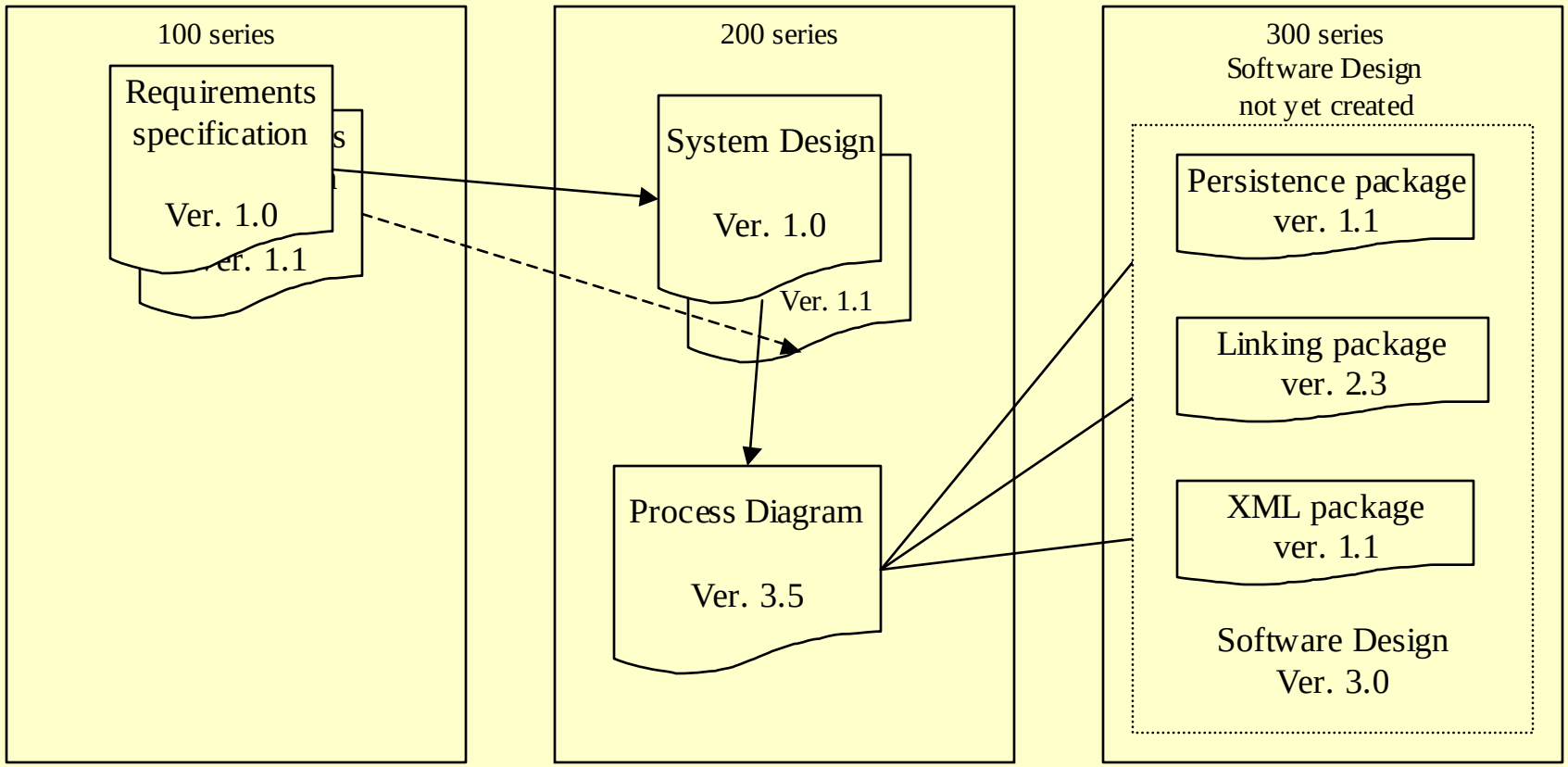
Design of the XML-ISDE

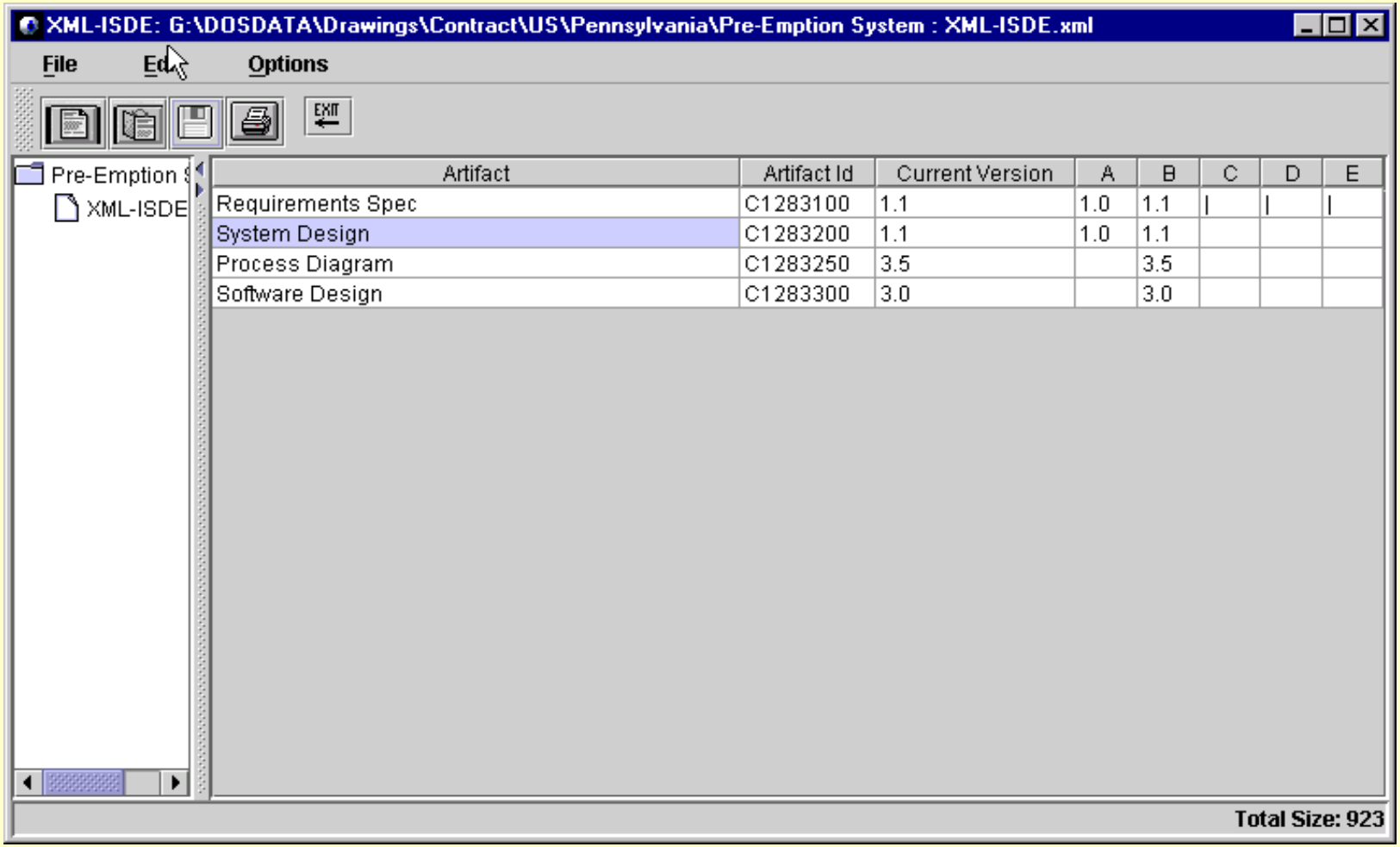


The eXtensible Control Table

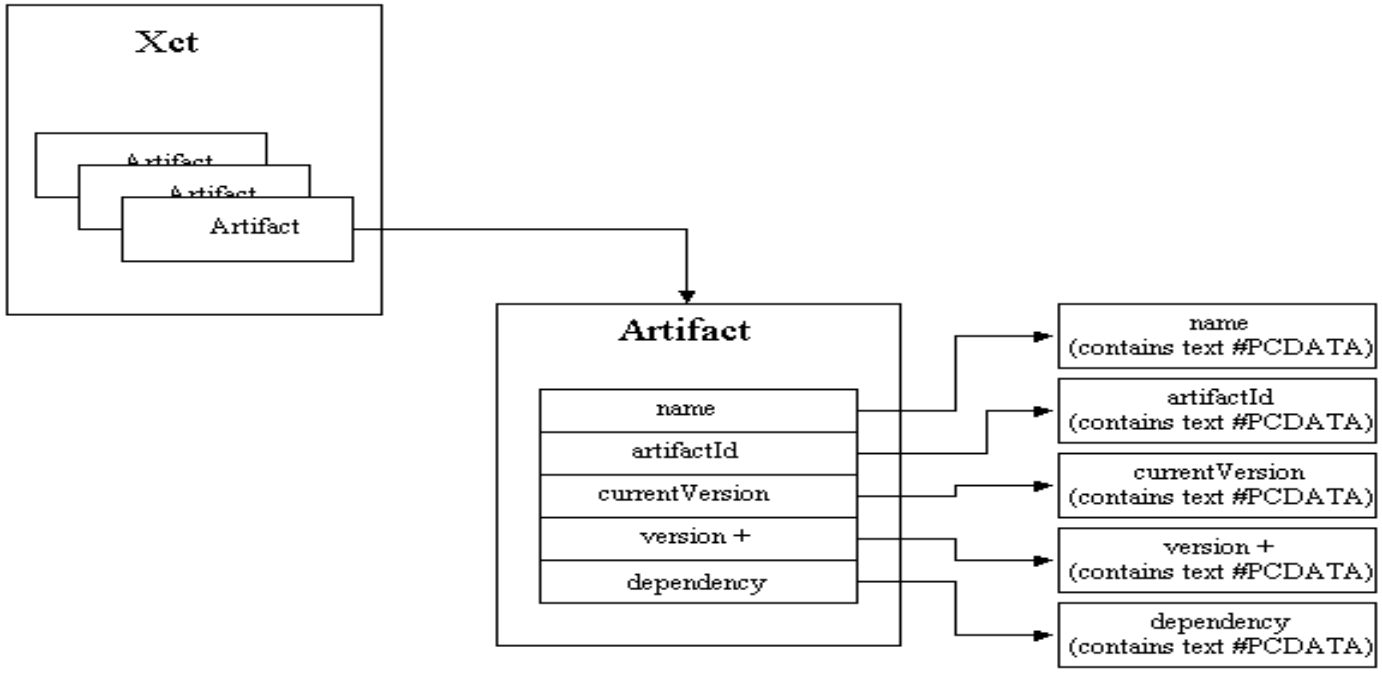
Configuration_Item Name: *XML Layer*		Configuration_id: 1283								
Artifacts	Artifact_id	Revision Level:								
		A	B	C	D	E	F	G	H	I
System design	128314	**1.1**	**1.1**	**1.1**	**1.2**	**2.1**				
Data management	1283112	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**				
Environment layer	12831103	**1.01**	**1.02**	**1.02**	**1.02**	**1.02**				
Presentation layer	12831106	**1.15**	**1.15**	**1.2**	**1.2**	**1.2**				

Implementation (create software design)

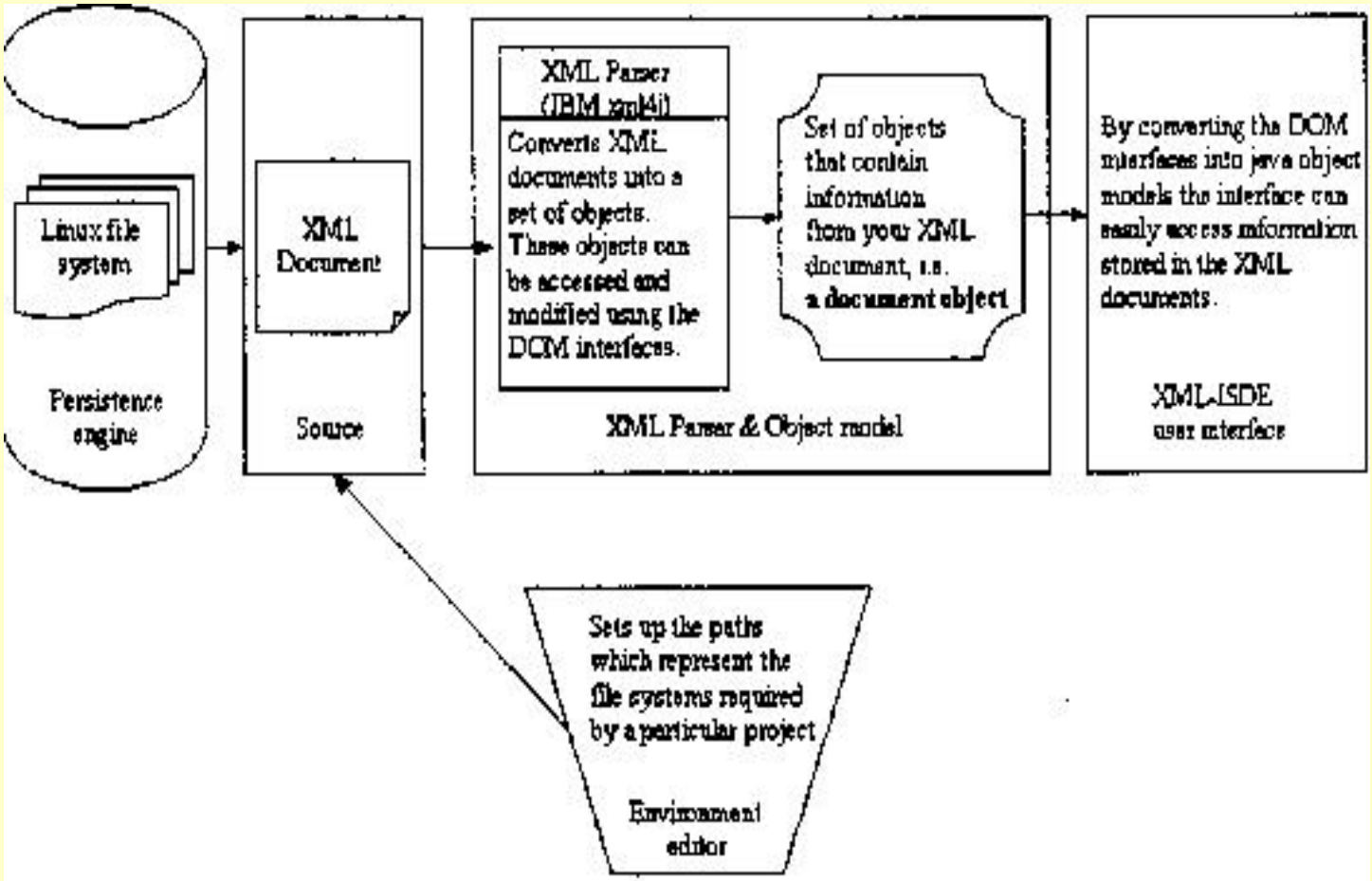



XML-ISDE: G:\DOSDATA\Drawings\Contract\US\Pennsylvania\Pre-Emption System : XML-ISDE.xml
File
Ed
Options
EXIT
Pre-Emption S
XML-ISDE
Artifact
Artifact Id
Current Version
A
B
C
D
E
Requirements Spec
C1283100
1.1
1.0
1.1
System Design
C1283200
1.1
1.0
1.1
Process Diagram
C1283250
3.5
3.5
Software Design
C1283300
3.0
3.0
Total Size: 923

DOM Tree Representation



Implementation Framework



Conclusions From Reagan's Project

- It is possible to use XML to create an improved documentation system linking the code domain with the document domain.
- It is difficult to cause the necessary organizational change to adopt the change.
- Stronger conviction than ever that software documentation must be treated as a complex whole.

Jeff's Project - Standardizing Documentation Using XML

- Documentation problems in small firms.
- Requirements for a documentation system.
- Why XML for documentation?
- Developing a document set.
- A document process.
- Extensions.
- Benefits and conclusions.

Small Software Firms

- Small software firms face many constraints.
- Documentation standards not a priority - no money to build standards and processes.
- Result is documentation that is disorganized, incomplete, and in many different formats.
- Maintenance problems are severe.

They need a simple, effective way of standardizing documentation

Requirements for a Documentation System

- Simple to install and use.
- Adoptable with minimum training.
- Extensible to different project types.
- Repeatable by different developers.
- Facilitate enforcing standards.
- Efficient in the use of time.

Why XML for Documentation?

- DTDs can enforce standardized format.
- Tools available – more under development.
- Flexibility through XSL for display and reporting.
- Extensibility as new needs arise.
- Open source industry standard, applicable across different platforms.

The Approach

- Based on experimental work in a software engineering class.
- Examined the (arbitrary) documents associated with a number of different software projects done in the past by the firm in question.
- Identified the documents that were common to most projects.

The Approach - continued

- Defined each of the recurring types of documents in terms of the sections they contain.
- Identified features, recurring sections, sequencing etc. Define this formally in the language of DTDs.
- By this means defined a basic set of documents to form the basis of a standard.

The Document Set

The documents selected cover three main areas of operation:

- Sales
- Project Management
- Development

Document Set - Sales

- RFP response.
  - Used to respond to RFPs. Explains the approach to the project, addresses specific sections from RFP.
  - Presents timeline and bid.
- Statement of work.
  - Indicates to client what work will be done in the project.
  - Serves as the basis of the contract between firm and client.

Document Set – Project Management - 1

- Task network.
  - Identifies all tasks and milestones in a project, along with effort estimates.
  - Indicates how images from packages should be included and versioned.
- Allocations.
  - Allocates team members to a given project.
  - Associates team member with task and time.
- Progress report.

  Reports the current status of the project, current issues and actions taken.

Document Set – Project Management - 2

- Meeting.
  - Identifies an agenda for a meeting, the proceedings from the meeting, and conclusions reached.
- Time entries.
  - Contains time entries for employees on a given project.
- Time entries report.
  - Generated from time entries document.
  - Summarizes time entries for a given project or employee, over a given time period.

Document Set – Development - 1

- Requirements.
  - Description of all the requirements for the project. Information comes from RFP and RFP response, and related meeting documents.
- Architecture.
  - Description of the framework and architecture of the system. Covers layering, database interactions and interfaces.
- UML package.
  - All UML diagrams and associated notes.

Document Set – Development - 2

- Data model.
  - Provides path to data model diagrams.
- URL scheme.
  - Defines the URL scheme for a web-based component of the project, Explains naming conventions and domain names used by the project.
- Directory structure.
  - Defines the file hierarchy of the components of the project in the file system.

Document Set – Development - 3

- Testing report.
  - Documents a test case, the results of executing the test case, and recommendations or bug reports.
- Presentation
  - Gives links to presentation materials in whatever form they were generated for each presentation. Consists of index, introduction, links and conclusion.
- Metrics
  - Presents the metrics for a project up to some given time. Most metric information is numeric.

Document Set Components

- Each document in the document set has the following associated components:
  - Fully documented DTD.
  - Working DTD.
  - XML template.
  - XSL templates as needed.

Documenting the DTDs

- Each DTD is documented as follows:
  - <tag_name> - Short tag description.
  - Content model – components or PCDATA.
  - Attribute definition.
  - Tag source -the namespace that the tag is from.
  - Element and attribute declarations.

Process for Documentation

- XML Document Generation
  - Templates - fast, helps learning curve
  - Tools - validating parser
  - XSL – formatting for various needs
- Document Access
  - Common repository
  - Browseable
  - Directory and URL structure
- Document Maintenance
  - Document set
  - Versions, guidelines

Extensions and Evolution

- DTDs need time and use to mature
  - Auto-generation tools
- XML DMS
  - Version control
  - Enforce file/URL structure
- Intelligent XML servers
  - Apache Cocoon, Java Servlet
- Good XML editor
  - Real-time validation - colour code tags, errors, etc

Benefits

- Promotes standardized documentation set and format
- Process-building easier
- Maintenance efficiency increased
- More professional documentation format
- Data mining potential (project histories)
- Faster production of new documentation
  - No formatting required
  - Structure pre-defined

Conclusions from Jeff's Project

- Provides a practical starting point.
- More experimentation is needed.
- The process is an important part.
- Pilot successful in a small firm.
- Are there similar projects ?
- Why not Docbook?

Judi's Project -APHID
Applying Patterns to Hypermedia Instructional Design

- APHID is an object-oriented approach to hypermedia design.
- Uses patterns for communication between developers.
- Semi-automatic generation of tutorials.
- Supports multiple types of instruction.
- Explicitly represents sequences and domain structure.

Patterns used by Aphid

Two types of patterns are used to guide the construction process:

- Presentation patterns represented as XSLT and CSS sheets.
- Instructional patterns represent the instructional plan :

  Review, Depth First, Spiral, Remedial.

Guidelines for Creating Hypermedia

Hypermedia design involves the same stages as software design.

- Data modelling.
- Navigational design modelling.
- Run-time behaviour model.
- User-interface design.
- Method for moving to implementation.

Technologies used in APHID

The software is a prototype drawing from a number of different technologies

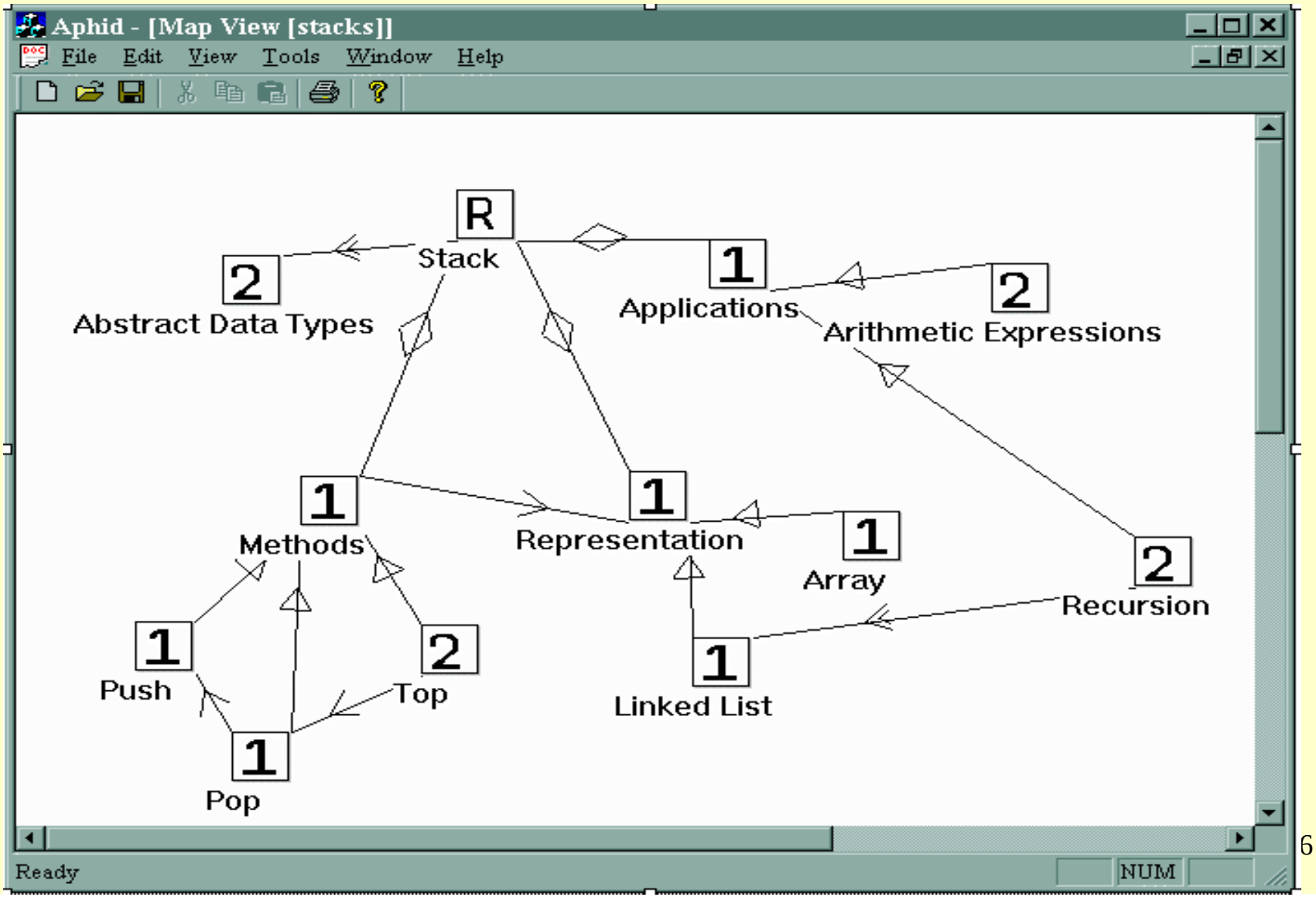
- The bulk of the software is written in C++ .
- The output is XML which is then formatted and ordered with XSLT.
- The output is then converted to HTML using an XSLT processing engine.

Creating an Application with APHID

Creating applications consists of 4 steps:

- Create concept maps.
- Create and assign data elements.
- Choose application parameters.
- Generate the application.

Concept Maps



Instructional Classes

Instructional classes are teaching structures i.e. quizzes, explanations, questions, or examples.

Instructional Class Document Types

```
<!--Quiz -->
<!ELEMENT quiz (description?, quiz_item+)+>
<!ELEMENT quiz_item (description?|(question,answer)+)>
<!ELEMENT question (#PCDATA)>
<!ELEMENT answer (#PCDATA)>
<!ELEMENT description (#PCDATA)>
    <!ATTLIST description
    type (intro|summary|conclusion|explanation|preformatted)
    "explanation">
<!--Question List-->
<!ELEMENT question_list (question_item)+>
<!ELEMENT question_item (question, answer?)>
<!--Instructions-->
<!ELEMENT instructions (description+, instruction_step+)>
<!ELEMENT instruction_step (description? |action)>
```

Document Type for Data Elements

```
<!ELEMENT data_element
    (title?, description?,
(
question_list|example|instructions|index|simulation|
practise_problem|quiz|FAQ|description|narrative
)
)>
<!ATTLIST data_element
id ID #REQUIRED
name CDATA #IMPLIED
importance (critical | explain | enrich) "explain"
level (beginner | intermediate | advanced) "beginner"
>
```

An Example Data Element

<list>

<description>The software is a PROTOTYPE drawing from a number of different technologies</description>

<list_item>The bulk of the software is written in C++</list_item>

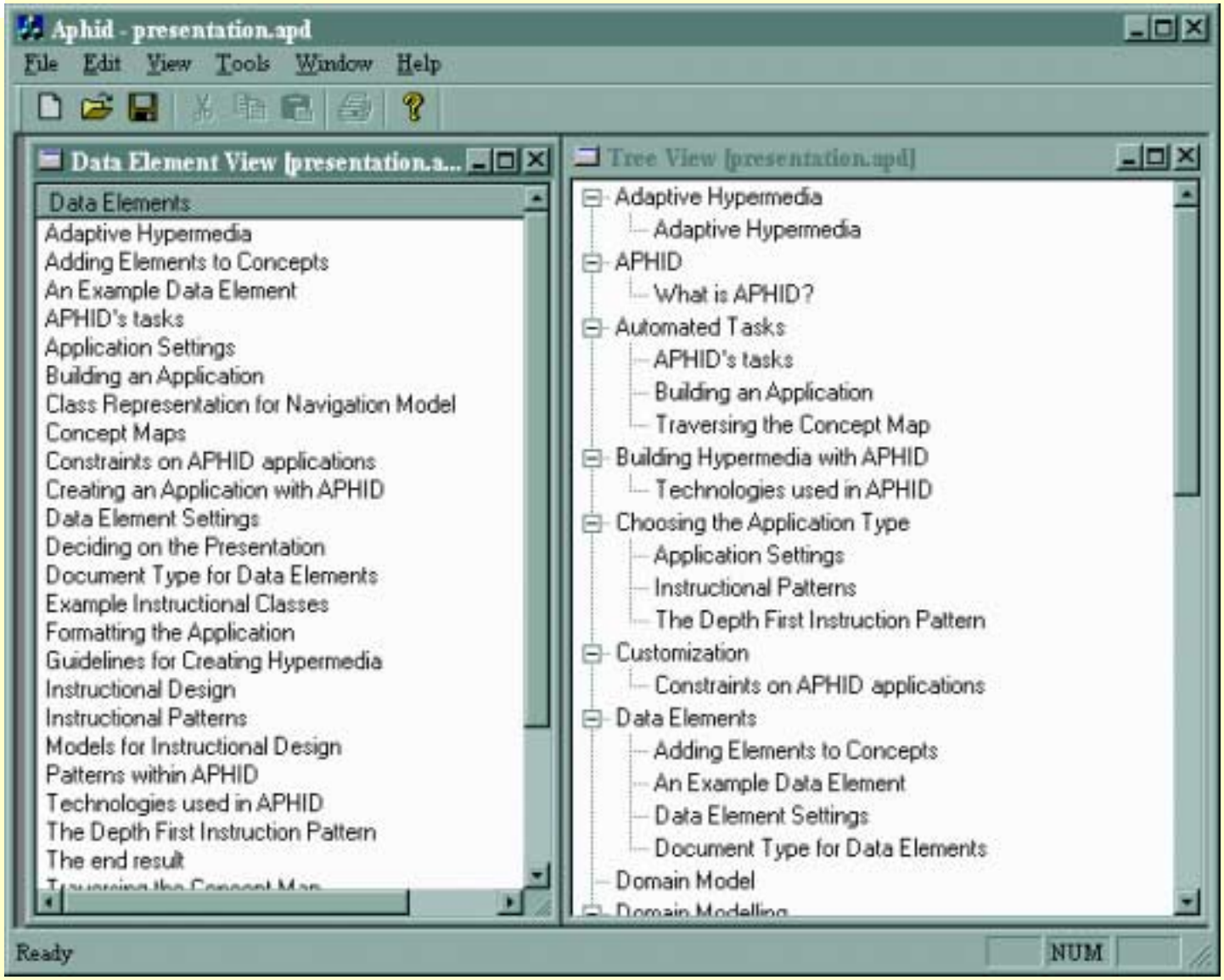
<list_item>The output is XML which is then formatted and ordered with XSLT </list_item>

<list_item>The output is converted to HTML using a java XSLT processing engine </list_item>

<list_item>currently James Clark's XT (with XP as the parser) </list_item>

</list>

Adding Elements to Concepts

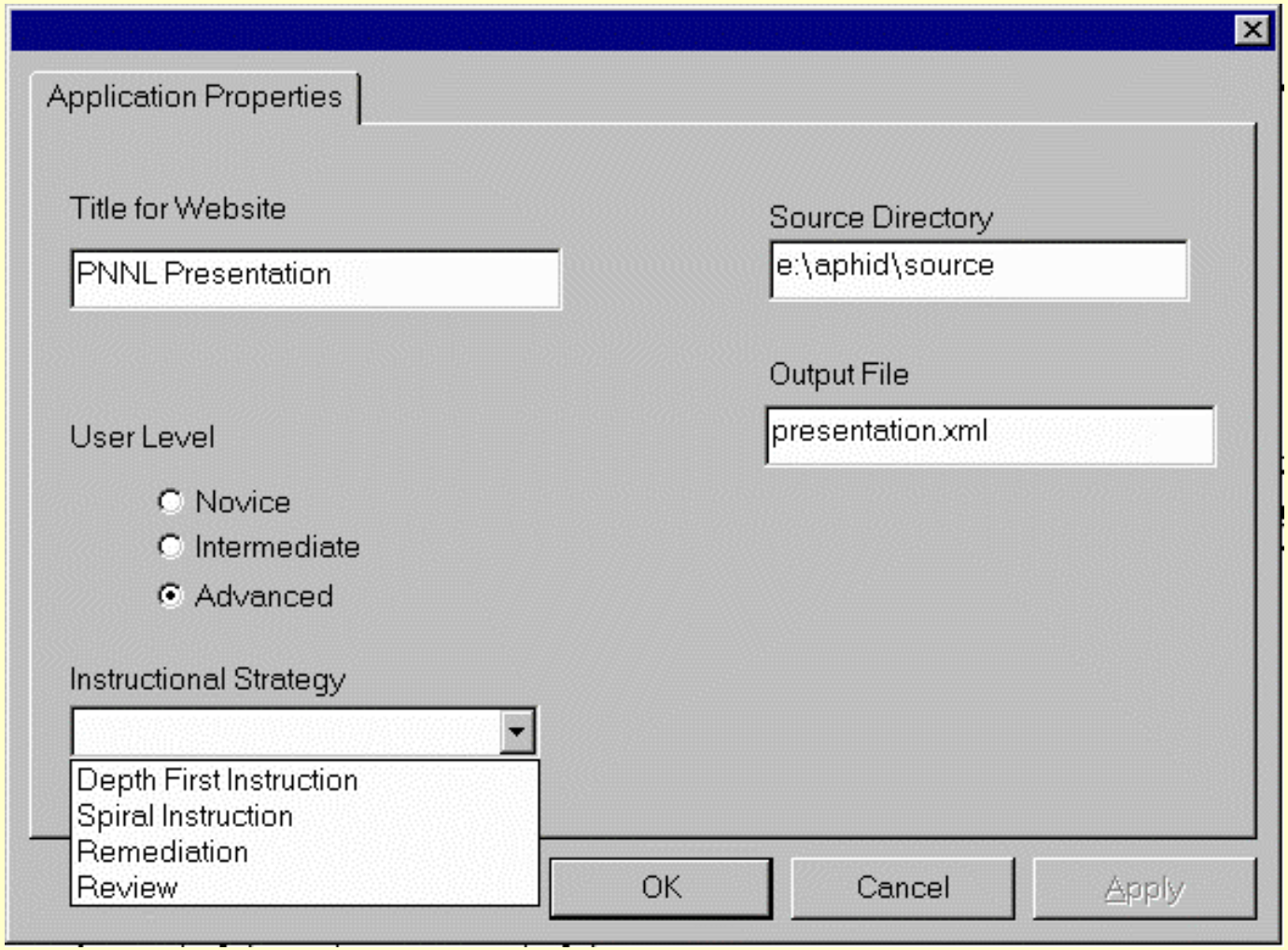


Choosing the Application Type

Instructional Patterns control several aspects of the final application.

- Which concepts are selected?
- The number of pages for a single concept.
- Tutorial-order for the navigational model.
- Which types of hyperlinks are selected?
- The order of the hyperlinks on a page.

Application Settings



Aphid's Tasks

- Select concepts.
- Build tutorial order (graph traversal).
- Select data elements.
- Check constraints and generate XML.

Building an Application

- Site generation starts with a traversal of the concept map.
- Two different traversal algorithms (depth/breadth)
- Infix, prefix, postfix and combo are possible.
- Result is a list of concepts in tutorial order.
- Data elements are selected for each concept in the tutorial list.
- Data elements selected based on type, difficulty, and level of user.
- Rules class specifies size constraints.

Deciding on the Presentation

- User Interface determined by XSLT and CSS.
- Every instructional class has at least one associated XSLT template.
- CSS governs font, colour, background.
- Easy to create sites that look and function differently from one another.

Final Result

- The result is a tutorial specifically generated to meet the conditions specified.
- Currently this consists of a set of web pages, linked together by hyperlinks.
- Many different tutorials can be generated from the same set of resources.
- Evaluations showed the method to be effective.

Conclusions from Judi's Project

Although the resources were used to generate web based tutorials they could just as easily be used for other end products, such as:

- Training materials.
- Help systems.
- Adaptable user interfaces, and.
- Software documentation.

Using XML for Effective Software Documentation

- What is the “ideal” solution to the problem of software documentation?
- A proposed approach using XML.
- What has to be done?
- Is it feasible today?

The Ideal Solution

- A unified approach to complete software documentation.
- All artifacts of a software system can be available through a single interface.
- Code, product documentation, process documentation, help and tutorials.
- Specific documentation generated as needed for a particular class of user.

Proposed Approach - XML

- XML enables a unified approach to software documentation.
- The ideal solution differs depending on firm, project, and users.
- XML can support the development of standard approaches suitable for many firms.
- XML can support customization to fit the needs of any specific firm.

Proposed Approach - 1

- Definition of a minimum set of documents or artifacts that comprise software documentation for the firm in question.
- Part of this set will be used for each project.
- Each document will have an associated DTD/Schema.
- XML wrappers will be used for non-textual documents, such as UML diagrams.

Proposed Approach - 2

- Each document type in the document set will have an associated XML template corresponding to the DTD/Schema.
- Generating each document type will be supported by a customized editor generating the necessary XML document.

Proposed Approach - 3

- Each document type will have a set of associated XSL style sheets supporting all formats in which the document may be needed.

Proposed Approach - 4

- All documents will be stored in a suitable repository/database.
- Versioning must be supported, along with updated relationships among documents.

Proposed Approach - 5

- Searching and browsing support will enable finding and retrieving particular documents.
- APHID-like techniques will be used to assemble combinations and sequences of elements from within the document set for help and tutorials.

What Has To Be Done?

- Determination of minimum document set by examination of past projects and views of developers.
- Development of XML support for editing, processing and storing of documents.
- Development of XSLs for document sets through analysis of user needs and problems.

Is It Feasible Today ?

- It is feasible today at the level of “proof of concept”.
- A system to do all of this could be created for a specific project or company but would be low on reusability.
- Further development of XML support tools is necessary to make it routine.

Where Will It Be Done ?

- By toolmaker firms?
- By large software development firms?
- By small software development firms?
- By all three types of firm?

The new and more effective CASE tool?