A Novel Update Propagation Module for the Data Provenance Problem:

A Contemplating Vision on Realizing Data Provenance from Models to Storage

Abed E. Lawabni, C. Hong, David H.C. Du, & *A. H. Tewfik*

Digital Technology Center, Intelligent Storage Consortium (DISC)

University of Minnesota

MSST 2005

April 12th, 2005

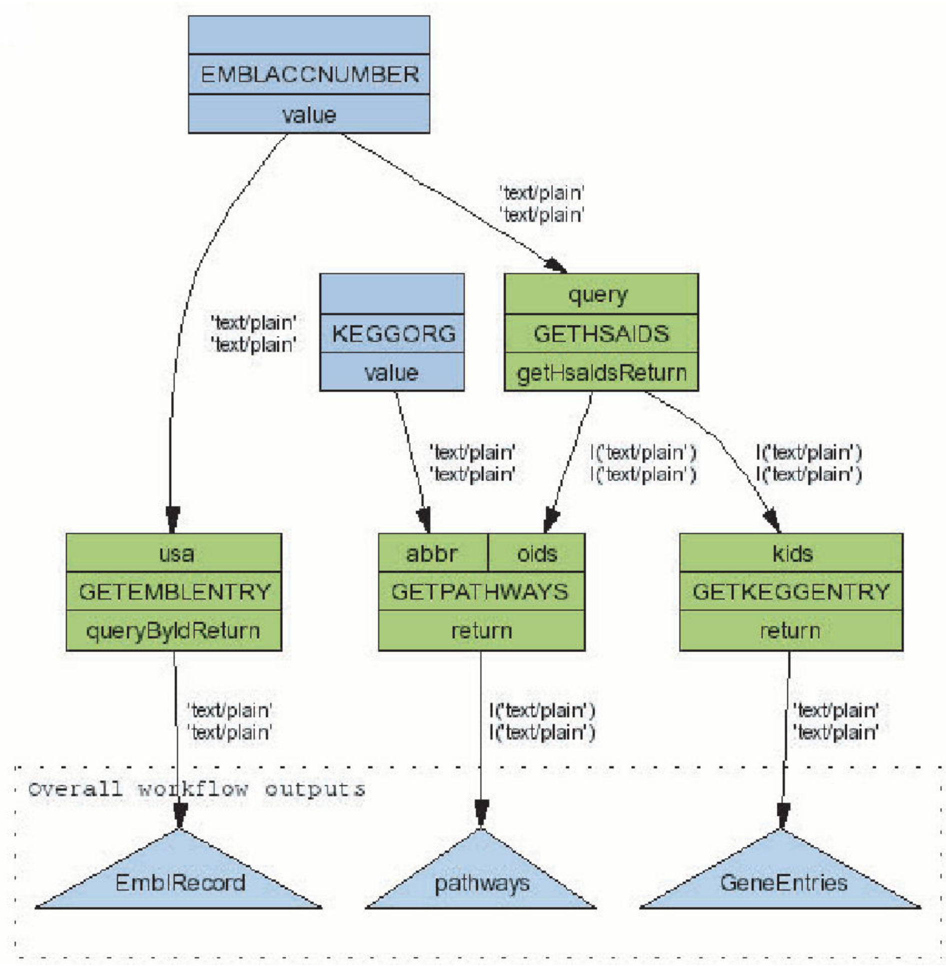
Outline

- Introduction & Motivation
- Automated Propagation Module
  - Sensitivity
  - Uncertainty
  - Complexity
- Compatibility with OSD technology
- Conclusions

Introduction & Motivation

Data Provenance

- ***Data provenance:*** is the description of the origins of a piece of data and the process by which it was generated
- The transition from laboratory science to *in silico* e-science (*computer-simulated experiments*) has facilitated a paradigmatic shift in the way we conduct modern science (e.g., bioinformatics, high-energy physics, etc.)
  - Experiments are designed as workflows containing processors and data links
  - Enactment of workflow is fully automated



Storage and Provenance

- Scalable storage systems can safely store and keep the data accessible for long time
  - In businesses, there are government or company regulations on how long certain data should be stored and be able to retrieved when requested
  - In scientific research, findings are build on past results and conclusions

- Unfortunately, little information on *where* and *how* a piece of data was derived, i.e., *provenance information*
- Lacking of provenance information undermines the usefulness of data:
  - Users doubt the *quality* and *reliability* of data

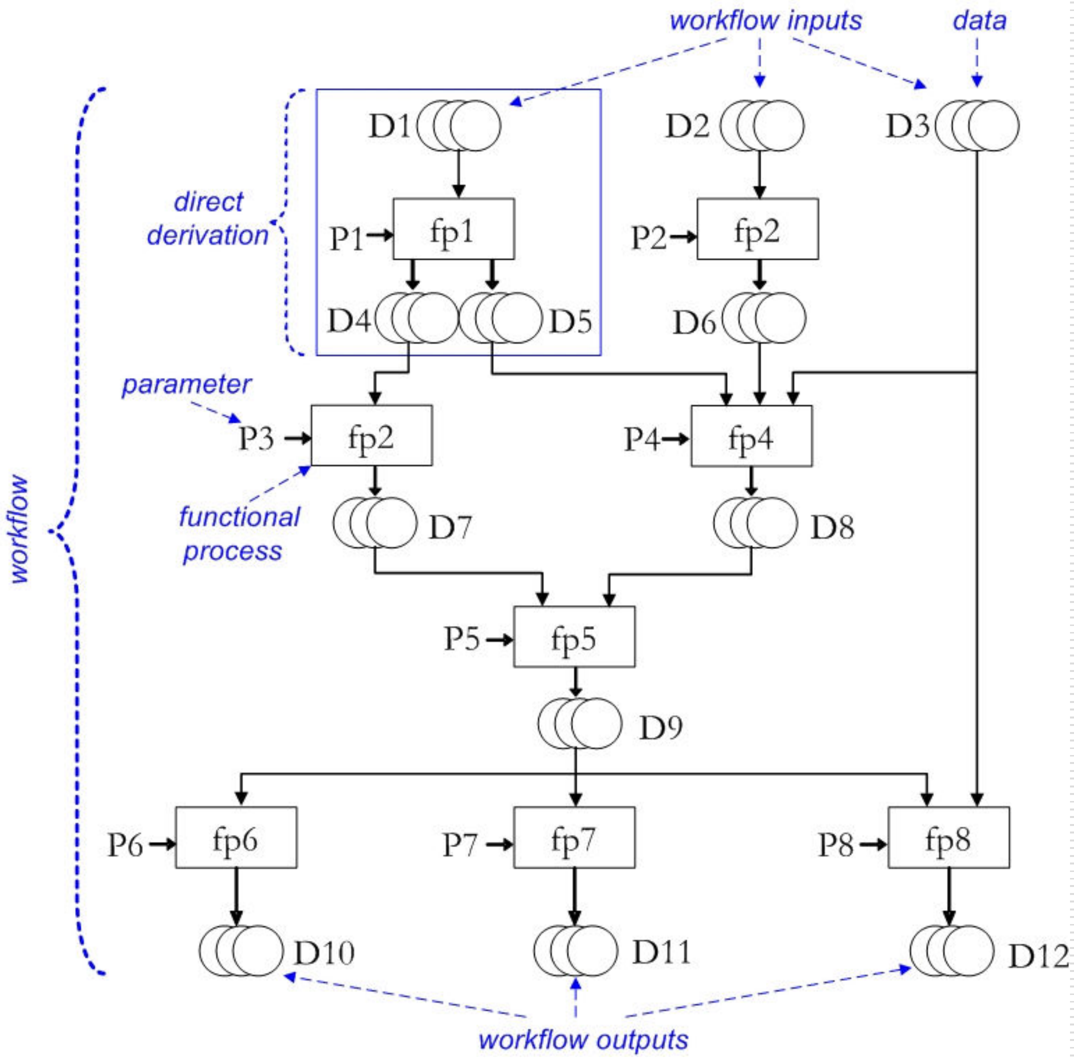
Limitations of Database-based Solution

- Existing approaches store all provenance records in a *central relational database* within the provenance server
  - One obvious limitation of this *centralized* solution is scalability. As more and more provenance information is recorded into the relational database, the overheads of performing access, queries and managements increase accordingly
  - When there are a lot of concurrent submissions of provenance records from concurrent clients, the provenance server becomes a bottleneck due to its limited processing power and buffer space
  - Data provenance relationship has to be in a pre-determined fixed format (database schema)
    - This may not be flexible enough for supporting multiple types of data objects and variable forms of data provenance relationships

Propagation Module

Data Provenance Model

- ***Direct derivations***: one-step processing of data
- ***Workflow***: a chain of direct derivations connected by using one's output as another one's input
- ***Functional process***: processing tools ranging from self-contained scripts/binaries to remote services
- ***Data***: permanently stored data not including temporary results within a functional process



What if ?

Data repeatedly copied/corrected/ transformed through numerous heterogeneous genomic database

- What if an update of a single input gives a full impact on some of the drawn conclusions?
- What if there is an error in one of the inputs or parameters?
- What if it takes tremendous computation time to re-run the whole experiment?

Objective & Approach

Objective: Automated propagation of changes, while

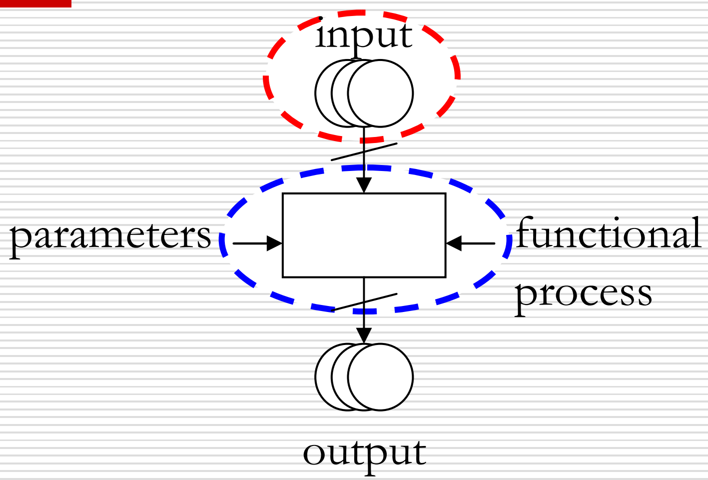
- Preserving all data as different versions
- Preventing the following scenarios:
  - Propagation of erroneous outcomes
  - Unnecessary rerunning of time consuming and heavily computations

Approach: Integrating three major decision factors as a ***sequential hypothesis testing problem*** to form a unique decision module

- Sensitivity analysis (*Variance-based Approach*)
- Uncertainty analysis (*Root-Sum-of the Squares Method*)
- Complexity

Basic Assumptions

- Two sources of errors:
  - Source data
  - Functional process



- Source data are *mapped* to *numerical values*
- Inputs can be correlated or independent

Sensitivity Analysis (SA)

- SA quantifies the effect of changing one or more input parameters on the output variability
- Two scenarios of interest:
  - Changes to source data (indep. & correlated inputs)
  - Minor mutation to the functional process

First Scenario: (Independent Inputs)

SA: Which input mostly contributes to the output variability ?

Model:

$$Y = f(X_1, \ldots\ldots, X_p)$$

Sensitivity indices are defined by:

$$S_i = \frac{V(E[Y|X_i])}{V(Y)}$$

$$\sum_{i=1}^{p} (S_i) = 1$$

[Sobol 1993]

First Scenario: (Correlated Inputs)

Consider the following Model: $Y = f(X_1, \ldots\ldots, X_p)$

Where

$$(X_1,\ldots\ldots,X_p) = (\underbrace{X_1}_{x_1},\ldots,\underbrace{X_i}_{x_i},\underbrace{X_{i+1},\ldots,X_{i+k_1}}_{x_{i+1}},\underbrace{X_{i+k_1+1},\ldots,X_{i+k_2}}_{x_{i+2}},\underbrace{X_{i+k_{l-1}+1},\ldots,X_p}_{x_{i+l}})$$

$(X_1, \ldots, X_i) = (x_1, \ldots, x_i)$ are indep. inputs

$(x_{i+1}, \ldots, x_{i+l})$ are correlated inputs

Sensitivity

$$S_j = \frac{V(E[Y|x_j])}{V(Y)} \qquad \forall j \in [1, \ i+l]$$

First Scenario: Correlated Inputs (continue)

If $j \in [1, \ldots\ldots, i]$, we have well define the same sensitivity indice:

$$S_j = \frac{V(E[Y \mid x_j])}{V(Y)} = \frac{V(E[Y \mid X_j])}{V(Y)}$$

And if $j \in [i+1, \ldots\ldots, i+l]$ for example $j = i+2$:

$$S_j = S_{\{i+k_1+1, \ldots\ldots, i+k_2\}} = \frac{V(E[Y \mid X_{i+k_1+1}, \ldots\ldots, X_{i+k_2}])}{V(Y)}$$

Second Scenario:
Impact of Minor Mutation of the Model

Model $$M : Y = f_1(X_1) + f_2(X_2, \ldots\ldots, X_p)$$

New Model $$M_{new} : Y^m = f_1(\mu_1) + f_2(X_2, \ldots\ldots, X_p)$$

where $\mu_1 = E[X_1]$

The new model sensitivity can be expresses by:

$$S^m = \underbrace{S}_{\text{Sensitivity of } M} \times \frac{V(Y)}{V(Y^m)}$$

Second Scenario: Impact of Another Type of Mutation of the Model

Two analysis have been made on two models:

$$M_1 : Y_1 = f_1(X_1, \ldots\ldots, X_p) \Longleftrightarrow S^1$$

$$M_2 : Y_2 = f_2(X_{p+1}, \ldots\ldots, X_{p+q}) \Longleftrightarrow S^2$$

$$M^{new} : Y^m = Y_1 + Y_2$$

$$S^m = S^1 \times \frac{V(Y_1)}{V(Y_1) + V(Y_2)} + S^2 \times \frac{V(Y_2)}{V(Y_1) + V(Y_2)}$$

Uncertainty Analysis (The Law of Propagation of Uncertainties)

Consider the following Model: $Y = f(X_1, \ldots\ldots, X_p)$

Suppose that each input X_i is associated with u_i uncertainty, then

$$u_{Y,i} = \left|\frac{\partial f}{\partial X_i}\right| u_i \qquad , (1 < i < p)$$

Sensitivity x *uncertainty*

Uncertainty Analysis
(Combining Uncertainties Due to Different Sources)

Root-Sum-of the Squares (RSS) Method:

Model $Y = f(X_1, \ldots\ldots, X_p)$

- The combined uncertainty of the **same** quantity (due to many different sources of uncertainties), say Y is given by:

$$u_{Y,combined} = \sqrt{\sum_{i=1}^{p} \left(u_{Y,i}\right)^2} = \sqrt{\sum_{i=1}^{p} \left(\left| \frac{\partial f}{\partial X_i} \right| u_i \right)^2}$$

Complexity

- Complexity denotes how long it will take to complete running a whole process. We will denote it by ***T***
- Intuitively a process with a high complexity should be more concerned with the other mentioned factors

All Together

- Once we obtain relevancy or sensitivity to changed/updated data and the degree of trust to associate with it, we pose the update problem as a ***sequential hypothesis testing problem***
- We assign the weighting factor corresponding to each decision factor
- Given a threshold obtained *empirically*, we evaluate if testing value from an update exceed this threshold

$$w_S S_Y^i + w_u u_{Y,combined} + w_C T \quad \underset{Do\ \ not}{\overset{Run}{\gtrless}} \quad \delta$$

weighting factors

Why OSD Technology?

Why Object-based Storage Technologies for Data Provenance?

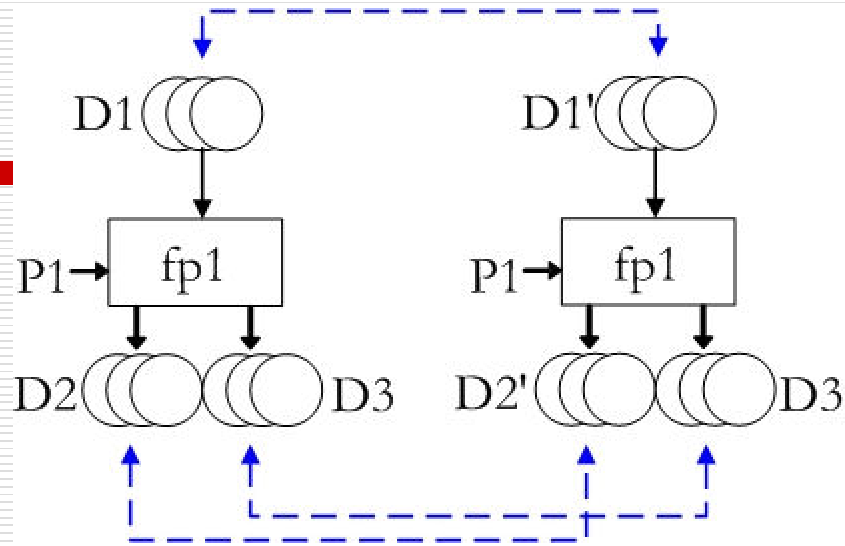
- Unique object ID (GUID)
  - Objects can be moved around w/o changing ID
- Extended attributes
  - Recording data object relationships and provenance information
- Highly-scalable distributed architecture
  - Compared to centralized solutions based on relational database or semantic web technologies
- flexibility :
  - No need to pre-determined fixed format (database schema)
  - Supporting multiple types of data object and variable forms of data provenance relationships

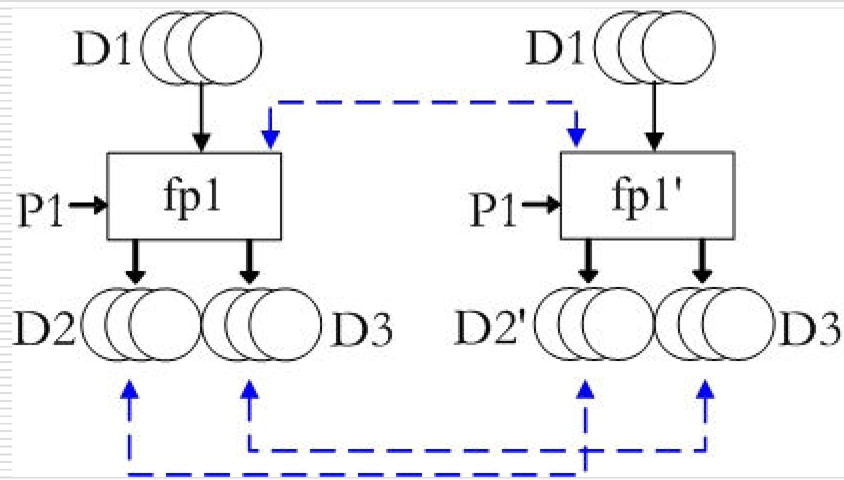
Versioning

- Functional process versioning
  - New processing tool perform the same function can be released. This requires explicit operation to set a newly-release processing tool as a new version of an older one
- Workflow versioning
  - A new version of one of its functional processes
  - Changes to parameters of its functional processes
  - Changes to the input data
  - Manually assigned by people. This requires workflow inputs and outputs have the same types

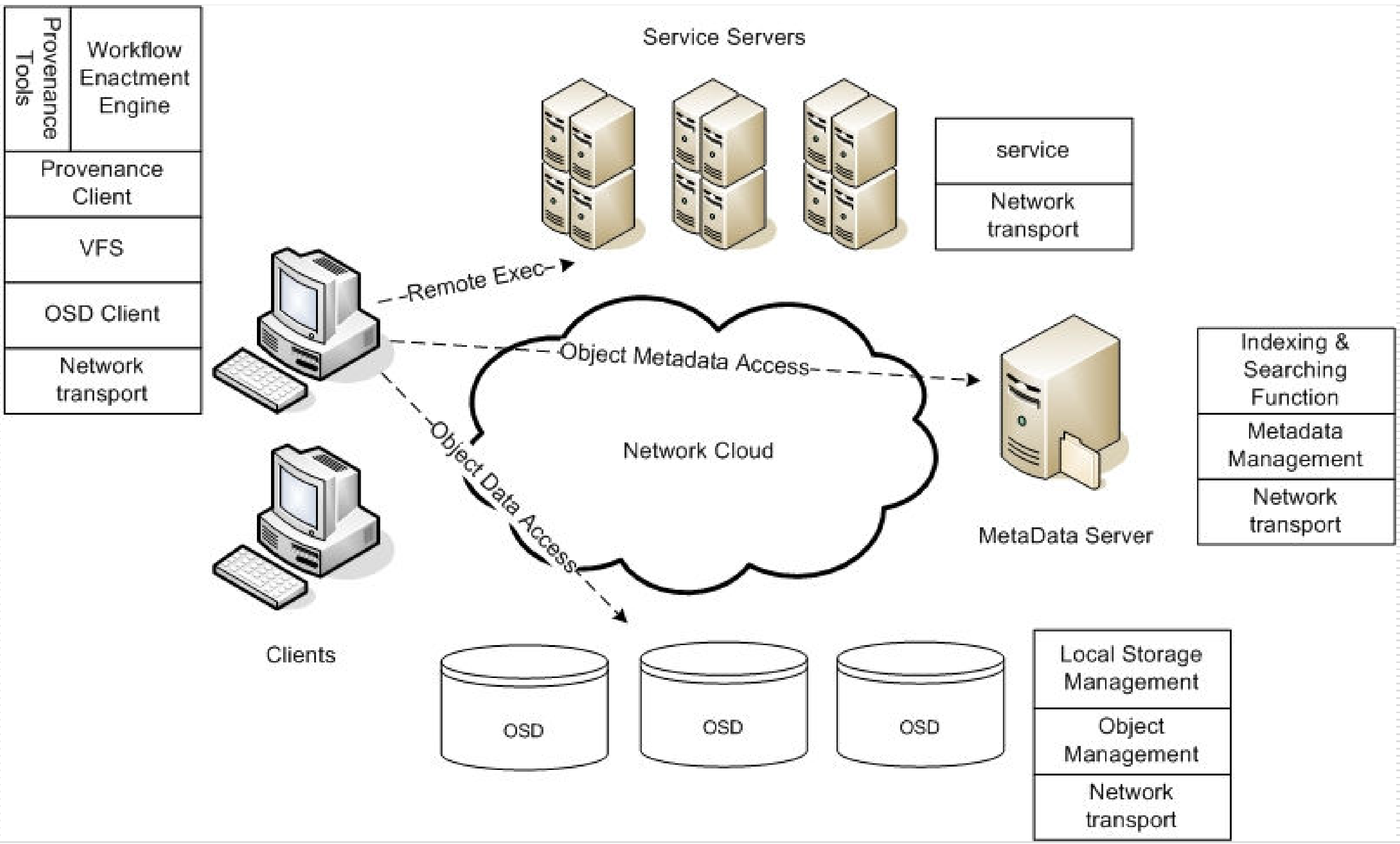
Versioning

- Data versioning:
  - Source data (from laboratory experiments) are manually assigned versions when new data is generated and stored
  - Derived data have new versions when new version of FP or new version of source data causes automatic propagation of changes



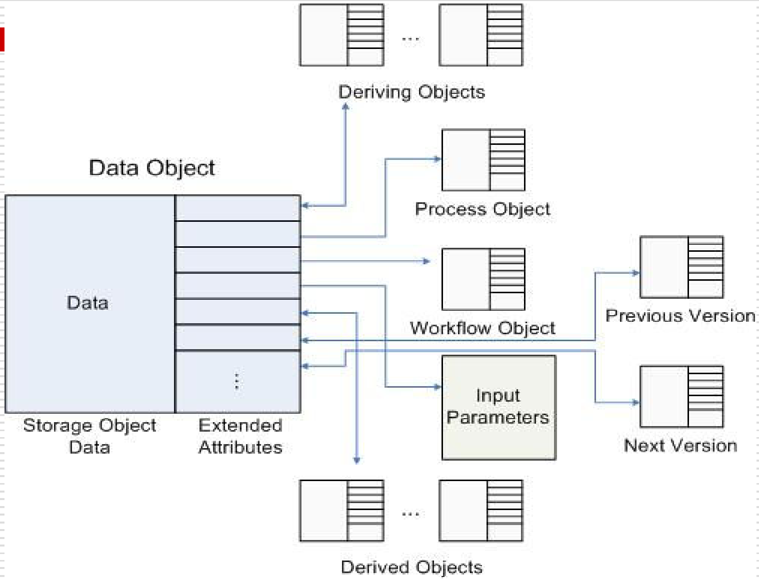


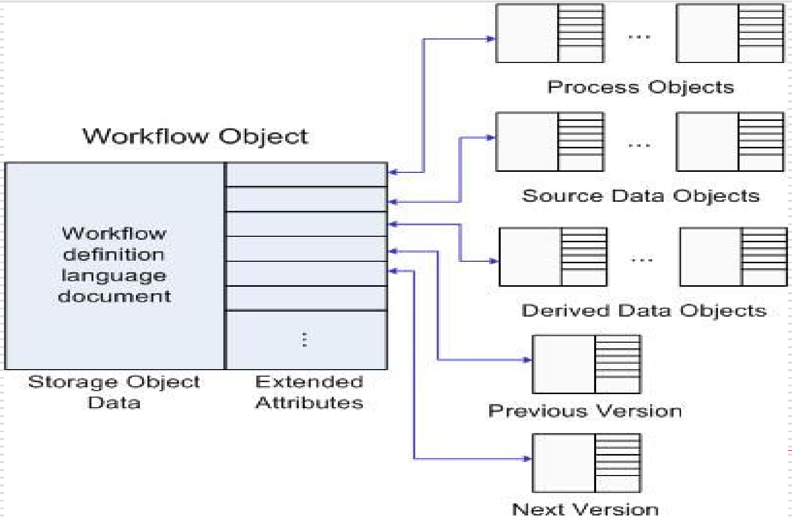
OSD-based Provenance System Architecture

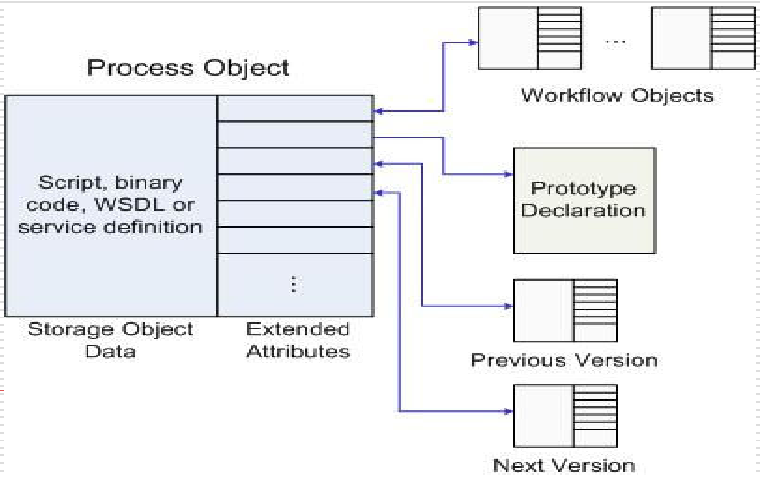


Provenance Objects

- Storage objects consists of data part and attribute part
- Data part is stored in OSTs
- Attribute part can be stored on either MDS of OSTs
- Provenance EA are stored in OSTs to take advantage of *active OSD feature*







Conclusions

- Automated propagation of changes caused by changed source data or changed processing tools
- All data are preserved as different versions
- A novel solution of data provenance using emerging object-based storage technology
- Highly-scalable distributed architecture compared to previous centralized solutions