



IBM Research

Simplifying Information Integration: Object-Based Flow-of-Mappings Framework for Integration

Howard Ho

IBM Almaden Research Center

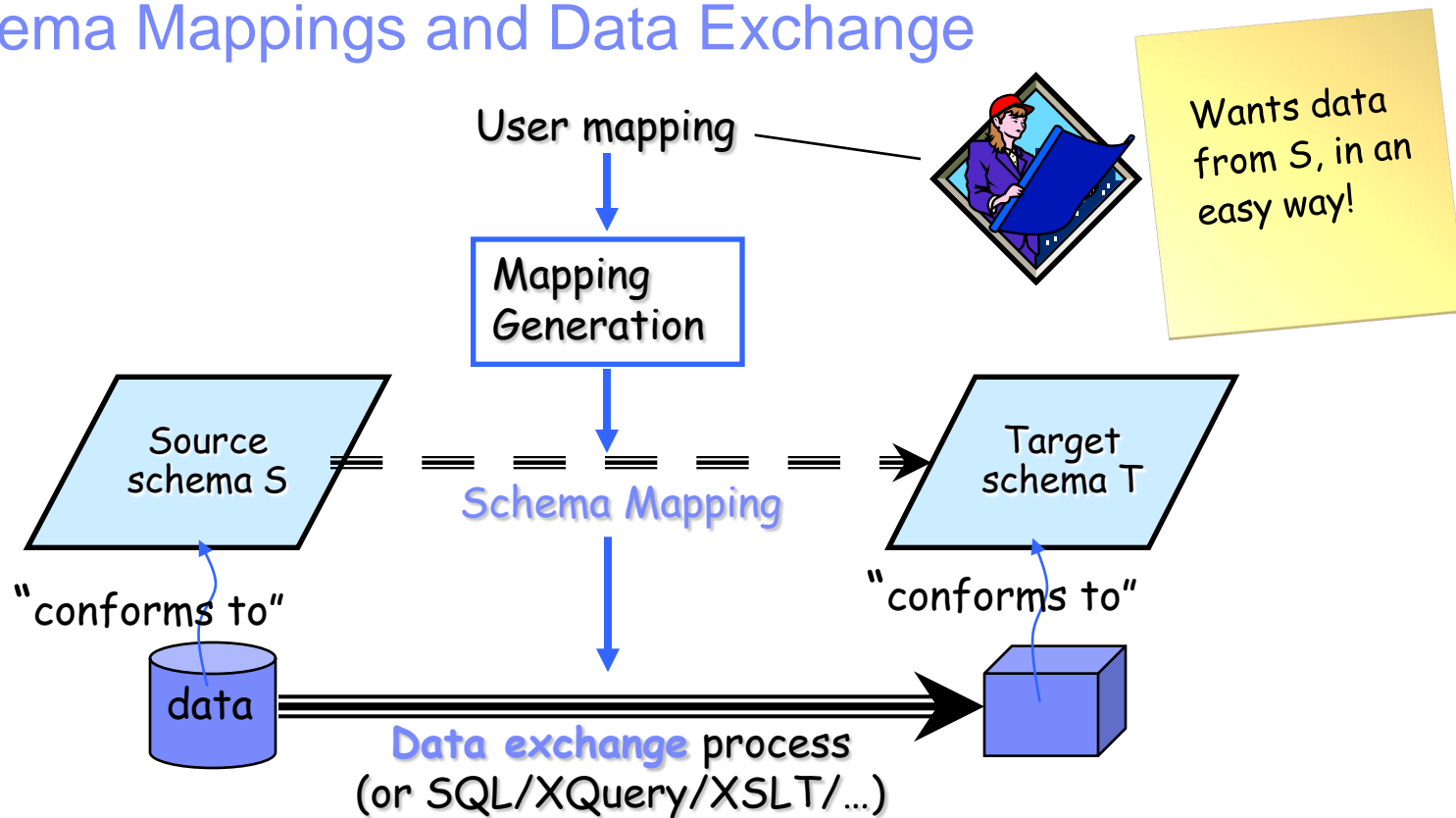
Take Home Messages

- **Clio (Schema Mapping for EII) has evolved to Clio 2010 (Flow of Mappings for ETL or Mashups)**
- **Raise the level of abstraction for data transformation (EII, ETL or mashups) to flow of mappings**
- **Compile flow of mappings for different runtime engines**
- **Unify Famous Objects in advance and incorporate into Clio 2010 to simplify integration tasks**
- **Need schema decomposition algorithms to do integration in Clio 2010 using UFOs**

Outline

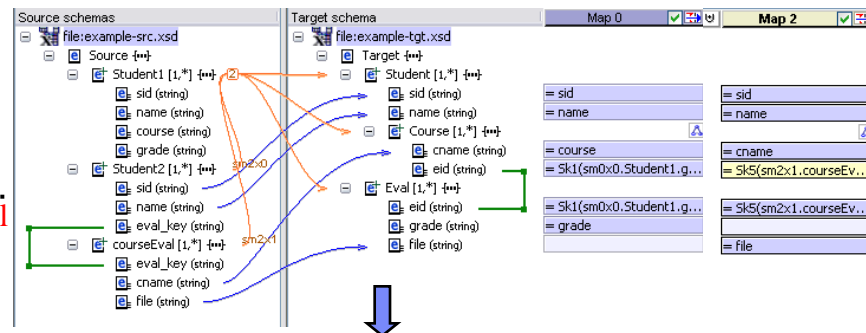
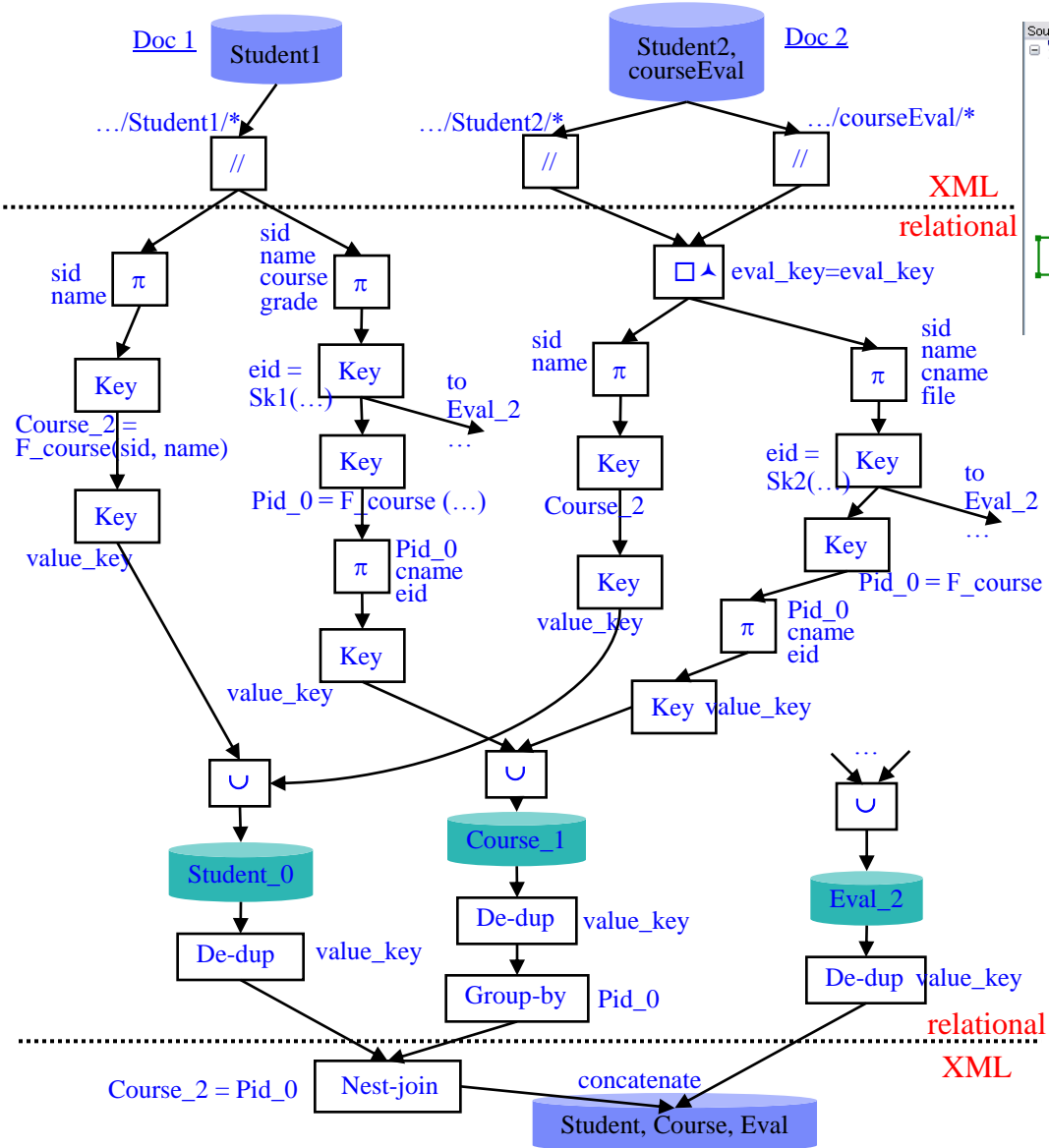
- **Clio Evolution**
- **Clio 2010**
- **Unified Famous Object (UFO)**
- **Schema Decomposition**
- **Joint work with**
 - Mauricio Hernandez, Lucian Popa, Ioana (Roxana) Stanoi
 - **Interns:** Bogdan Alexe (UCSC), Michael Gubanov (UW), Jen-Wei Huang (NTU, Taiwan), Yansis Katsis (UCSD), Barna Saha (UMD)

Schema Mappings and Data Exchange



- Schema mappings are "bridges" for information integration
- Our research (*Clio* project) addressed two main problems:
How to **generate schema mappings** and how to use them for **data exchange**?

Clio: From Mappings to Runtime Artifacts (Queries or Operators)



```

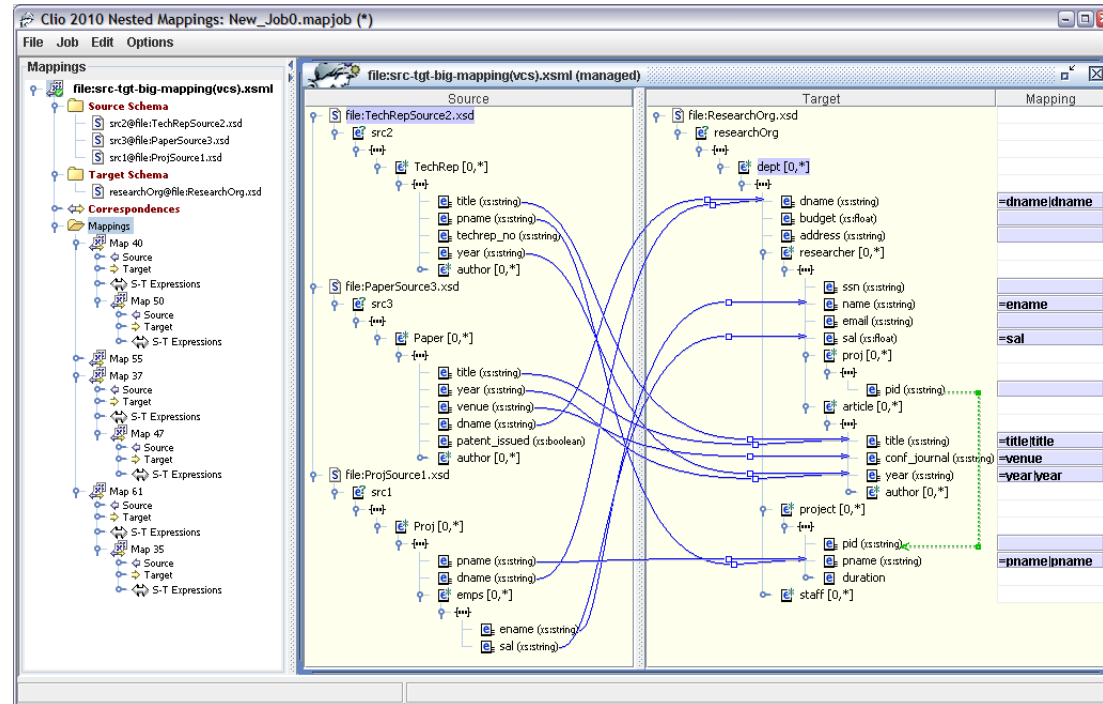
(===== Phase-1: shredding into relational views =====)
declare function local:shred($doc0 as element(root)) as element(root) {
  <root>
  {
    for $sm0x0 in $doc0/Source/Student1
    return
      <Student_0>
      <sid_0>{ $sm0x0/sid/text() }</sid_0>
      <name_1>{ $sm0x0/name/text() }</name_1>
      <Course_2>{ fn:concat("F_course(", $sm0x0/sid/text(), ", ", $sm0x0/name/text(), ")") }</Course_2>
      <value_key>{ fn:string-join(($sm0x0/sid/text(), $sm0x0/name/text(), fn:concat("F_course(", $sm0x0/sid/text(), ", ",
      $sm0x0/name/text(), ")"), ")") }</value_key>
      </Student_0>
  }
  {
    for $sm2x0 in $doc0/Source/Student2,
    $sm2x1 in $doc0/Source/courseEval
    where $sm2x0/eval_key = $sm2x1/eval_key
    return
      <Student_0>
      <sid_0>{ $sm2x0/sid/text() }</sid_0>
      <name_1>{ $sm2x0/name/text() }</name_1>
      <Course_2>{ fn:concat("F_course(", $sm2x0/sid/text(), ", ", $sm2x0/name/text(), ")") }</Course_2>
      <value_key>{ fn:string-join(($sm2x0/sid/text(), $sm2x0/name/text(), fn:concat("F_course(", $sm2x0/sid/text(), ", ",
      $sm2x0/name/text(), ")"), ")") }</value_key>
      </Student_0>
  }
  {
    for $sm0x0 in $doc0/Source/Student1
    return
      <Course_1>
      <Pid_0>{ fn:concat("F_course(", $sm0x0/sid/text(), ", ", $sm0x0/name/text(), ")") }</Pid_0>
      <cname_1>{ $sm0x0/course/text() }</cname_1>
      <eid_2>{ fn:concat("Sk0(", $sm0x0/course/text(), ", ", $sm0x0/name/text(), ", ", $sm0x0/sid/text(),
      <value_key>{ fn:string-join((fn:concat("F_course(", $sm0x0/sid/text(), ", ", $sm0x0/name/text(), ")"), $sm0x0/course/
      text(), fn:concat("Sk0(", $sm0x0/grade/text(), ", ", $sm0x0/course/text(), ", ", $sm0x0/name/text(), ", ", $sm0x0/sid/text(),
      </Course_1>
  }
  }
}
    
```

Evolution of Clio

- **Nine years of active research**
- **Created and established the research area of schema mappings and data exchange**
 - 14 patents and 22 database PIC papers, 7 papers invited to special issues of journals
 - 5 of these papers have received 200+ citations each (254, 213, 198, 166, 143)
 - 1 paper (composition) won 2004 Pat Goldberg Memorial Best Paper Award.
 - Many invited talks, 4 keynotes, 2 tutorial
- **The Clio system has evolved to be a design-time tool (based on mappings) that can “represent” multiple runtimes:**
 - Query execution platforms: SQL, SQL/XML, XSLT, XQuery
 - Java
 - Data Flow engines: ETL (DataStage and E2), mashups (DAMIA), [now JAQL flows on Hadoop](#)
- **Transfer to IBM products**
 - DB2 8.1 SQL Assist
 - DB2/II 8.2 XML Wrapper Generator (Chocolate)
 - IBM Content Manager 8.3 (Cinnamon)
 - Rational Data Architect v7 (Criollo)
 - Information Server v8 (FastTrack and BackTrack)

Limitations of the Clio Mapping Paradigm (pre-2007)

- Based on XML/relational schemas
- Based on one, **monolithic**, source-to-target mapping
 - Building and managing complex mappings becomes difficult.**
- The idea of flow of transformations was missing.
- The mapping process lacked modularity and reusability

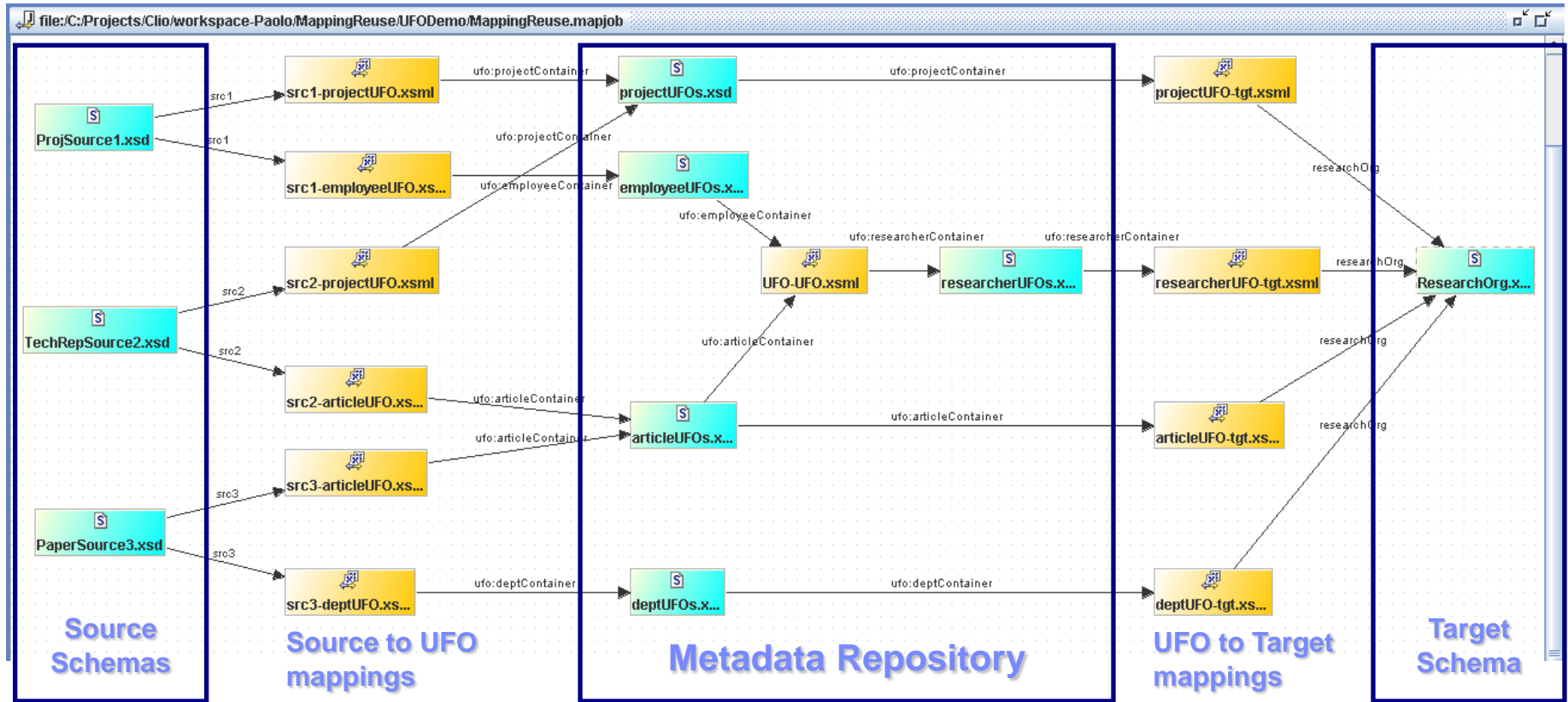


- Needed to be adaptive to more data models (Java objects, JSON) and runtimes (DAMIA, Hadoop -- Cloud Computing)
- Needed to raise the level of abstraction (business objects – UFOs)

Clio 2010: Declarative Flows of Mappings

- Vision: Schema mappings are building blocks (steps) in a flow
 - Clio 2010: extension of Clio where multiple mappings can be defined, loaded, reused and managed
 - The graph of mappings is a declarative specification of the flow of data
- Additional Semantic Data Model: UFOs (semantic objects)
- Novel mapping technology in Clio 2010: automatic assembly of a graph of initial, uncorrelated mappings into larger, richer mappings
 - Includes *mapping composition, mapping merge (correlation)*
 - *Correlation*: Join and fuse data coming from the individual mappings
- Deployment: Richer flow of mappings can be compiled/optimized into a global execution plan
 - Queries, transformation scripts, ETL flows, Map/Reduce (e.g., Hadoop), Mashups (e.g. DAMIA)

Clio 2010: Flows and UFOs



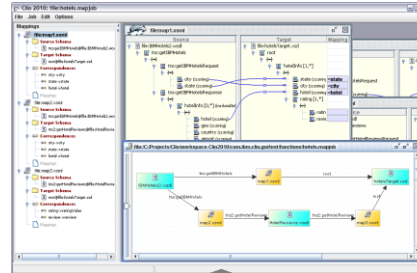
Key points:

- modularity and simplicity
- higher-level of abstraction
- reusable pieces

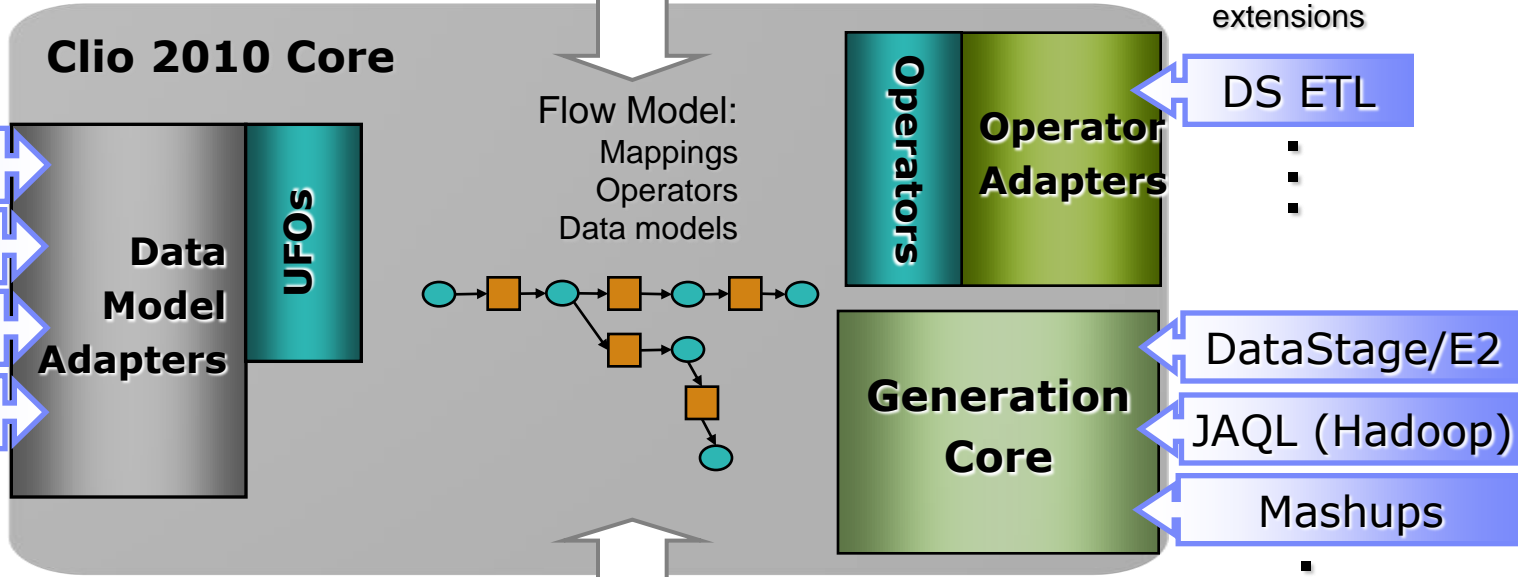


Clio 2010 Architecture

GUI

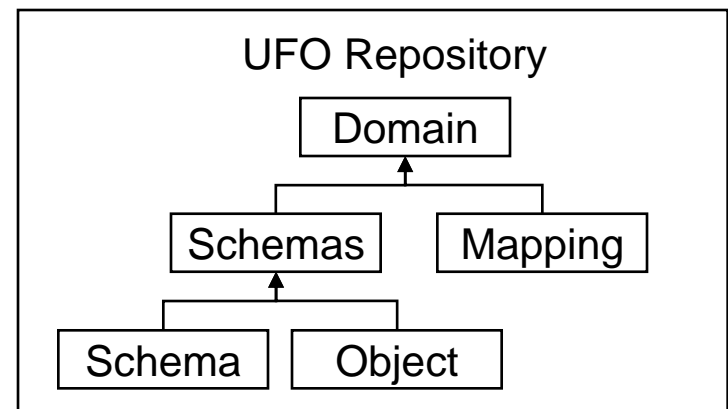
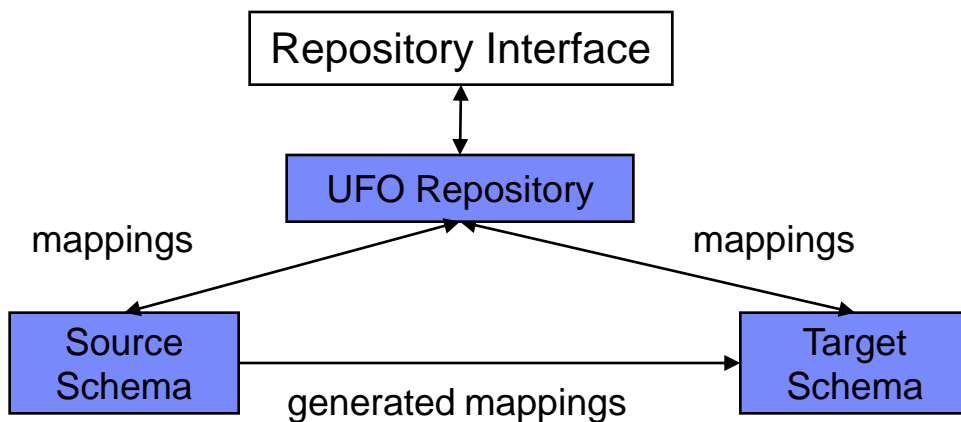


extensions



UFO Model

- Unified representation of famous objects
 - Decompose complicated concepts into simple flat objects
- Predefined relationships between UFOs
 - Make the schema mapping job easier for users
- UFO repository interface
 - Allow users to browse, query and modify UFOs and mappings

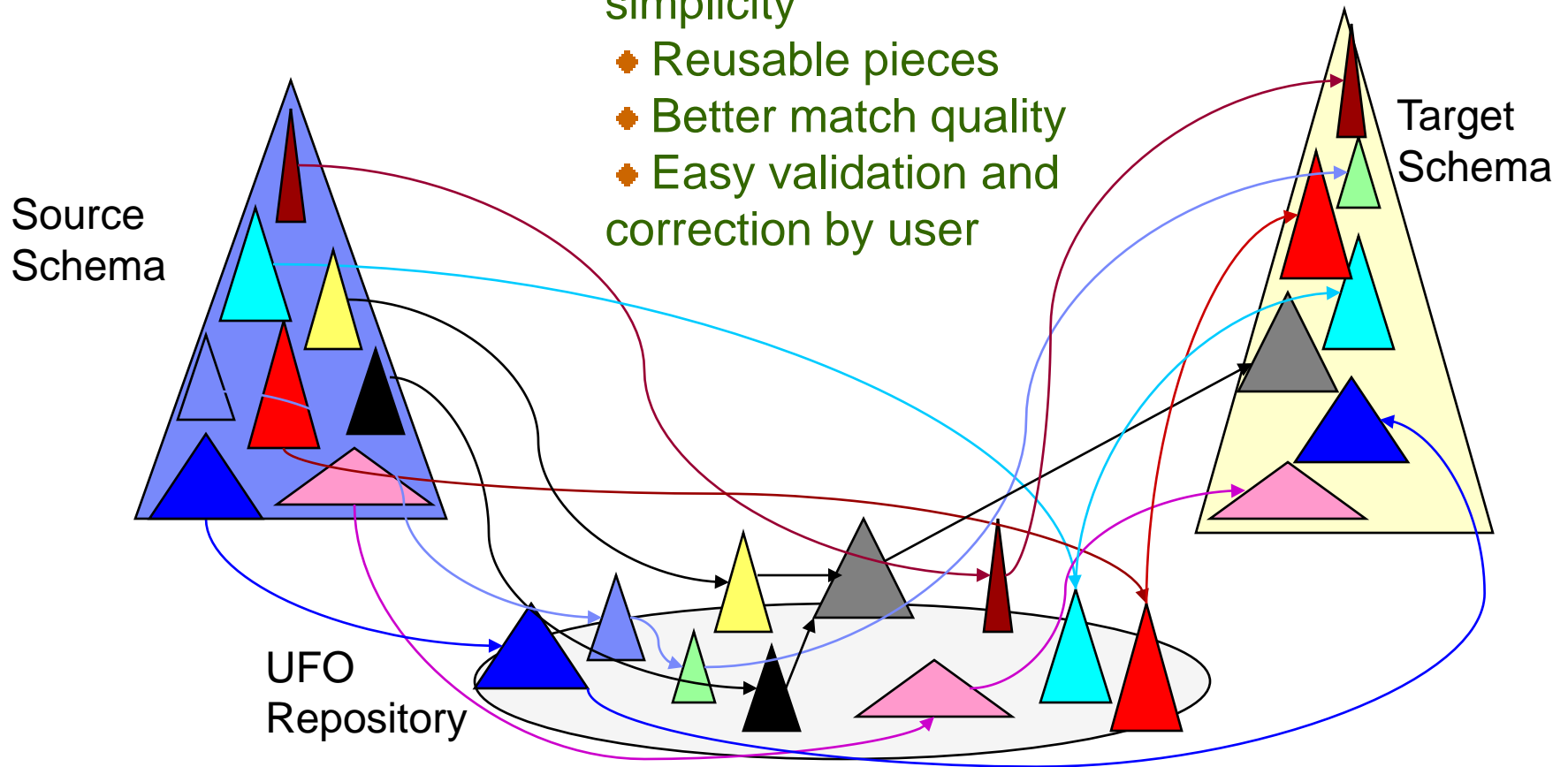


* each data contains two extra fields:
created user and URI

Schema Matching with UFOs

Advantages:

- ◆ Modularity and simplicity
- ◆ Reusable pieces
- ◆ Better match quality
- ◆ Easy validation and correction by user



Schema Matching with UFOs

- **A 2-step technique**
 - **Step 1:** *decompose Source and Target*
 - **Step 2:** find paths between Source and Target through UFOs
- **Currently we focus on Step 1**

Schema Decomposition

Summary of Schema Decomposition

- **Use a three-step approach for solving the schema decomposition problem using UFOs.**
 - **Step 1: Filtering**
 - Filter out irrelevant famous objects based on high-level signatures of schema and UFOs
 - **Step 2: Similarity Score Computation**
 - Calculate actual similarities only for reduced search space
 - **Step 3: Obtain Maximum Coverage**
 - Compute coverage (which UFOs to use and where)

Goals: Clio 2010, UFOs and Schema Decomposition

- **Establish Clio 2010 as useful design-time abstraction behind many runtime platforms that do data transformation**
 - ETL, Mashups, Cloud Computing
- **Precisely define the UFO data model.**
 - Use Freebase, OAGIS, and industrial models (SAP) as sources of UFOs
- **Use Schema decomposition to match schema and UFOs**
- **Goal in 2008**
 - Demonstrate an integration scenario using flow of mappings based on UFOs
 - Demonstrate Clio 2010 generating transformation code for different platforms
 - E.g., **JAQL queries for JSON data & Map/Reduce**, DAMIA scripts, **E2**, DS ETL.



Unified data integration environment

- Object based, common and open data model, mapping-based optimization, compiles to parallel engine via Map/Reduce