Object Versioning and Information Management

DR. SOUHEIL KHADDAJ

FACULTY OF COMPUTING, INFORMATION SYSTEMS AND MATHEMATICS
KINGSTON UNIVERSITY

EMAIL: s.khaddaj@kingston.ac.uk
Summary

• Overview
• OO and Information Management.
• Object Versioning.
• Case Study.
• Conclusion
Overview

• Temporal information systems using object orientation for
  – Examining the pattern of continuous changes of attributes and behaviour object over time
  – Tracking of evolution of object over time
  – Mapping the model to object oriented database
Object Orientation and Information Management

- Object orientation allow real world objects to be well represented in information systems.
- It encourages modularity within information systems.
- It improves the ability to create complex objects.
- It provides the extensibility needed to create new models.
- New software can be developed from existing ones. Thus, providing reusability.
Data Models

• **Relational Model**
  – inability to represent real world object and behaviour
  – inefficient data management
  – inadequate object relationships handling
  – can’t handle complex structure and relationships of objects
• **Object Oriented approach**
  – represent real world objects: abstraction
  – manage application data properly: encapsulation
  – handle object’s relationships efficiently: aggregation, association
  – manage complex structure and relationships: Inheritance and polymorphism
Object Versioning

• Changes of objects are handled using version management:
  – Starting with a generic object
  – First and subsequent changes can be represented as versions.
• Each version of the object reflects changes of attributes and/or behaviour.
• Subsequent changes of the versions will generate:
  – related dynamic attributes
  – and temporal links to be updated to respective versions.
• The version management uses temporal operators (e.g. during, after, before etc) to handle gradual and sudden changes.

• Version management reduces the need of large storage space.
  – Only the generic object or the current object holds the complete attributes and behaviour of the object.
  – Other versions represent the changes of their attributes and behaviour.
Complete Versions

- $\text{Versions}(x) = (CV_x(n), CV_x(n-1), \ldots, CV_x(n0))$
- $CV$ represents the complete version, $n$ indicates the number of the version, $x$ is the object and $n0$ is the oldest version number.
- Each version can be accessed by reference to the number of the version, $n$.
- Although access to any version is supported directly and all versions have similar access time, storage space can be costly.
Linear versioning

\[ V_0 \rightarrow V_1 \rightarrow \ldots \rightarrow V_{n+1} \rightarrow V_n \]

- \( V_0 \) = Generic version,
- \( V_1 \ldots V_n \) = Versions of the generic object
- \( \rightarrow \) = Temporal topology link
Forward Linear Versioning

• Generic object holds the complete attributes and behaviour
• The temporal relationships between the generic object and versions is given by:

\[ \text{Versions}(x) = (\Delta_x (n,n-1), \Delta_x (n-1,n-2), \ldots, \Delta_x (n_0+1, n_0), CV_x(n_0)) \]

- \( CV_x(n_0) \): complete version of object \( x \) whiles \( n_0 \) indicates the generic version.
- \( \Delta_x (k, k') \): represent changes between the current version and the previous version of object \( x \).
• Faster access for older versions
Backward Linear Versioning

• Current object holds the complete attributes and behaviour
• The temporal relationships between versions is given by:

\[ \text{Versions}(x) = (CV_x(n), \Delta x (n, n-1), \Delta x (n-1, n-2), \ldots, \Delta x (n0+1, n0)) \]

\(CV_x(n)\): complete version of object \(x\)
\(\Delta x (k, k')\): represent changes between the current version and the previous version of object \(x\).

• Faster access for newer versions
Branch Forward Versioning

\[ V_0 \rightarrow V_{1,0} \rightarrow V_{1,1} \rightarrow \cdots \rightarrow V_{1,m} \]
Branch Backward Versioning

\[ V_{n-1,0} \rightarrow V_{n-1,1} \rightarrow \ldots \rightarrow V_{n-1,m} \rightarrow V_n \]
**Forward versioning:**
- Generic object holds the complete attributes and behaviour
- The temporal relationships between the generic object and versions is given by:
  $$\text{Versions}(x) = (\Delta_x (n, n_0), \Delta_x (n-1, n_0), \ldots, \Delta_x (n_0+1, n_0), CV_x(n_0))$$

**Backward versioning:**
- Current object holds the complete attributes and behaviour
- The temporal relationships between the current object and versions is given by:
  - $$\text{Versions}(x) = (CV_x(n), \Delta_x (n,n-1), \Delta_x (n,n-2), \ldots, \Delta_x (n, n_0))$$
• Different versioning strategies might be more suitable for different applications.
• Linear versioning: attributes changes and/or behaviour changes.
• Branch versioning:
  – Forward for splitting objects.
  – Backward for merging objects.
• Combined strategies.
Combined strategies

\[ V_0 = \text{Generic version (parent)}, \]
\[ V_n = \text{Object merging} \]
\[ V_{1,0} \ldots V_{1,m} = \text{Object splitting}, \]
\[ V_{1,0} \ldots V_{t,n} = \text{Linear Versioning} \]
\[ \ldots = \text{Temporal topology link} \]
Geographical Information Systems (GIS)

- **Temporal GIS**
  - Identify, store and examine specific geographical changes over time
  - Changes in maps

- **Problems**
  - Snapshot
    - space, time consuming and costly to maintain
    - Inefficient handling continuous changes over time
OO Model for GIS

Composite classes of a geographical object
UML design showing semantic of the versioning and temporal relationships
• Temporal OO GIS Model
• UML diagram
• Object Oriented Programming Environment (OOPE); C++
• Object Oriented Database System (OODBS) based on Objectivity/DB
Case Study - Land Registry

Spatio-temporal domain in 1930:
1: Grass land
2: Road
3: Forest

Spatio-temporal domain in 1940:
1: Grass land
2: Barren land
3: Agriculture Site

Spatio-temporal domain in 1980:
1: Flood land
2: Dual Carriage
3: Golf Course

Spatio-temporal domain in 1990:
1: Airport site
2: Motorway
3: Race Course
4: Road
Figure 13: Example of links between versions, events and processes of geographical changes
Other applications

Composite classes of a document object
Conclusion

- The versioning approach provides an efficient technique for continuous tracking of the evolution of objects.
- Less data storage will be required since only the changes to the objects (versions) are stored.
- Object oriented database eliminates the mapping error experienced in implementation of temporal IS systems.