

SML 95-0239

***FOREST:
Software Configuration Management in
an Object Oriented Database***

*Mick Jordan
Michael L. Van De Vanter*

USENIX Conference on Object-Oriented Technologies
June 28, 1995

The Software Development Environment (SDE) Challenge

Size and complexity increase

- **systems**
- **teams**
- **language/host/target environments**
- **demand for *global* information**

Current tools do not scale

- **weak configuration management**
- **old tools stretched beyond original design limits**
- **new tools limited by inadequate infrastructure (directories, files, pickles)**
- **ad hoc “databases” in file system**

Current OODB Technology Can Help

Precise modelling of SDE data

- **persistent typed objects**
- **small atomic objects and large structured objects**
- **highly interconnected**

Robust sharing

- **client-server**
- **transactions**

Using OODB Technology: Key Design Issues

Scalability

- **control database size**
- **avoid contention**
- **model systems precisely**

Extensibility

- **support new object types**
- **integrate tools**

Usability

- **assure acceptable performance**
- **access to tools**

Forest Foundations

ObjectStore[®] OODBMS

- **transacted, persistent C++ objects**
- **client-server model supports shared virtual memory**
- **other DBMS features not used**

The *Vesta* approach

- **repository of immutable components: *source* and *derived***
- **systems organized as versioned *packages***
- ***system model* expressed as a functional program**

ObjectStore and Object Design Inc. are registered trademarks of Object Design Inc.

Forest Overview

Forest integrates

- **configuration management**
- **editing**
- **system building**

Forest data management

- **extensible framework based on Component class**
- **exploits characteristics of the application**
- **cooperates with system builder**

Forest Overview (cont.)

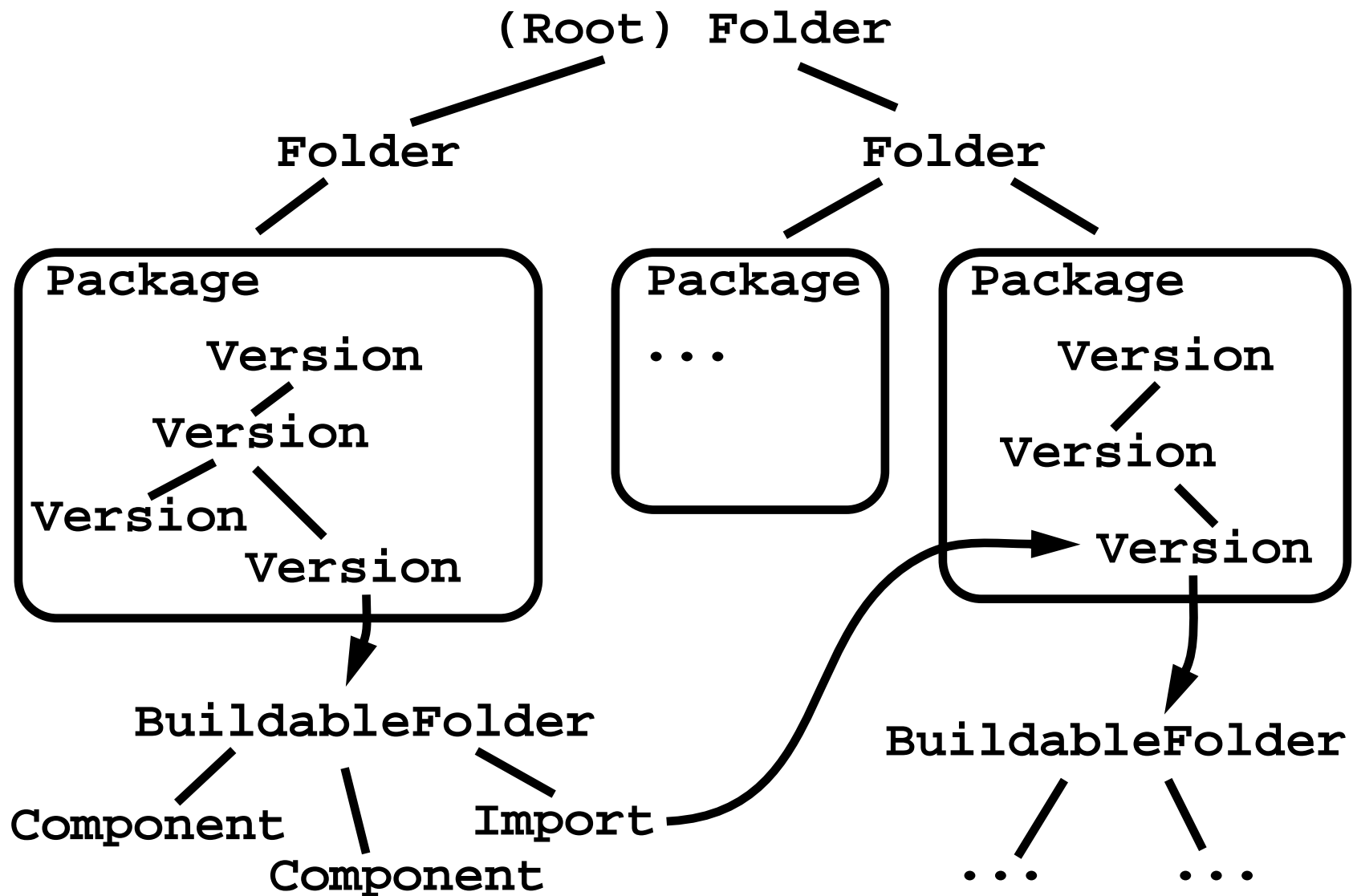
The Forest Project

- **ObjectStore database shared by a group of people**
- **cross-project *links* are possible**

The Forest Component

- **abstract base class for all persistent objects**
- **128 bit UIDs**
- **mostly immutable**
- **reference counted**
- **abstract value (identity - represented with a *fingerprint*)**
- **DBNewPlace class abstracts placement strategy**

A Forest Project (Database):



Forest Storage Management

General approach

- **exploit immutability of components**
- **encapsulate an extensible placement strategy**
- **tune strategies to avoid contention**
- **manage “hot spots” carefully**

Placement mechanisms

- **DBNewPlace class reifies placement strategy**
- **a Component method that returns its DBNewPlace**
- **implemented using ObjectStore placement mechanisms**

Forest Storage Management (cont.)

Basic policies

- **minimize conflict - maximize concurrency => short transactions**
- **use ObjectStore References to hold browsing state**
- **default allocation: same placement as containing component**
- **separate segment for each package**
- **separate segment for derived components**

Garbage collection

- **garbage components arise in several ways**
- **Component layer supports GC by reference counts**
- **creating/removing *links* is a distributed transaction**

Controlling Database Size

How much to store persistently?

- **derived information dominates storage**
- **immutability and precise configuration management permit caching and sharing**

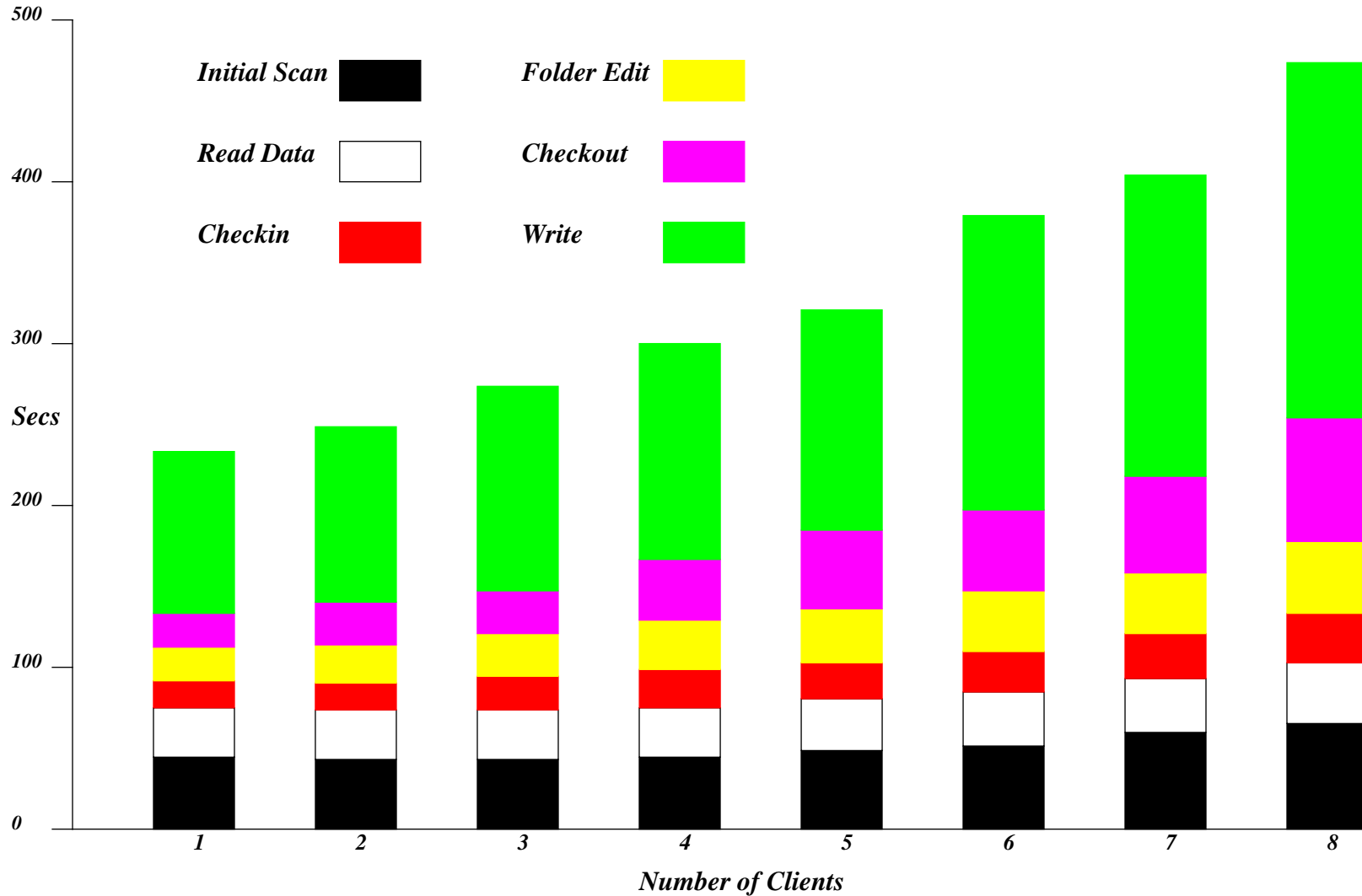
Separation of interface and implementation

- **access Components through abstract interfaces**
- **each implementation makes a time-space trade-off**
- **example: *Annotated Abstract Syntax Tree***

Conflict Avoidance

- **Mostly solved by immutability of components**
- **UID data structures are a hot spot**
 - **direct consequence of distributed shared memory model**
 - **applies to any global resource with high update rate**
- **Solve by distributing UID space among components**
 - **define a UIDGenerator interface**
 - **any component can inherit and implement UIDGenerator**
 - **UID generating components form a tree structure**
- **Exploit expected access patterns**
 - **make each package a UID generator**
 - **could extend this to more levels if needed**

Benchmark Results (Multiple UID Generators)



Conclusions

Component objects are a sound basis for an SDE

- **strong typing**
- **immutability**
- **separate interface and implementation**

Current OODB technology works

- **relatively simple programming model**
- **transacted data is well worth the cost**
- **object clustering is important (together and apart)**

Integration is important

- **versioning, editing, and system building**

Status and Future Plans

Prototype

- **versioning system implemented and stable**
- **component-based compiler for Clarity C++**
- **system modeler for Clarity C++ programs**

Build a working prototype for mainstream languages

- **e.g. C++, Java, Modula-3**

Explore further use of fine-grained components

- **language-dependent build-avoidance**
- **structured document editors**
- **configuration management for HTML**

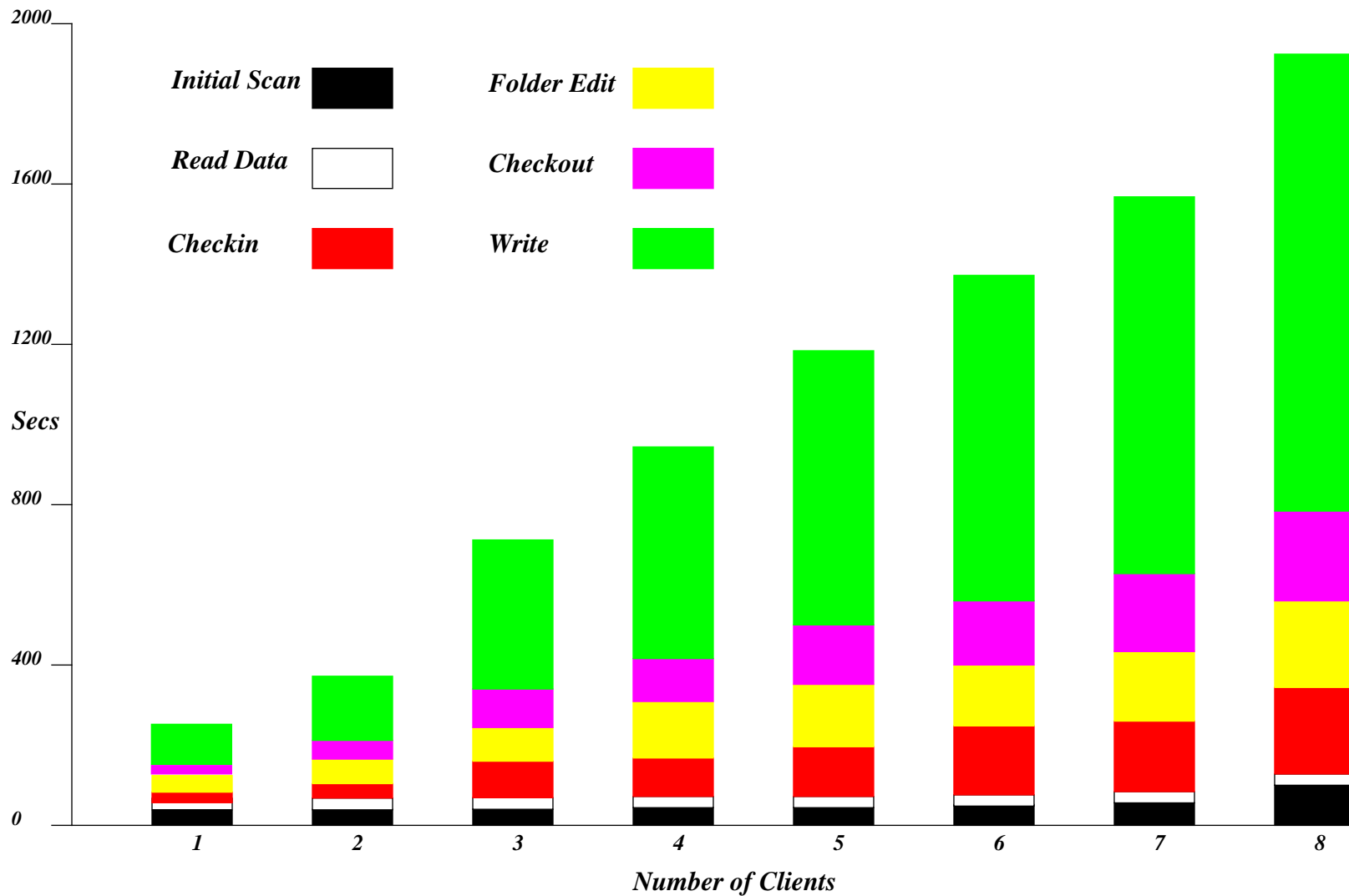
BuildableFolders

A BuildableFolder supports precise configuration management and system building

It can be thought of in three ways:

- **A *folder* of source components**
 - like a file system directory
- **A *compound document***
 - the unit of versioning
- **A *program module***
 - evaluated to build derived components

Benchmark Results (Single UID Generator)



Versioning System Benchmark

How does performance scale as users are added?

- **simulates a user performing a sequence of operations:**
 - checkout a package at random
 - select a random number of components to edit
 - edit each component (by making a copy)
 - checkin new package version
- **each operation is a separate transaction**
- **2500 text files (C++ source) organized into 216 packages**
- **initial database size is 18Mb**
- **25 iterations**
- **multiple users run on separate machines with different random number seed**
- **64Mb dual-processor SPARCstation™ 10**

SPARCstation is a trademark of SPARC International Inc., licensed exclusively to Sun Microsystems Inc.