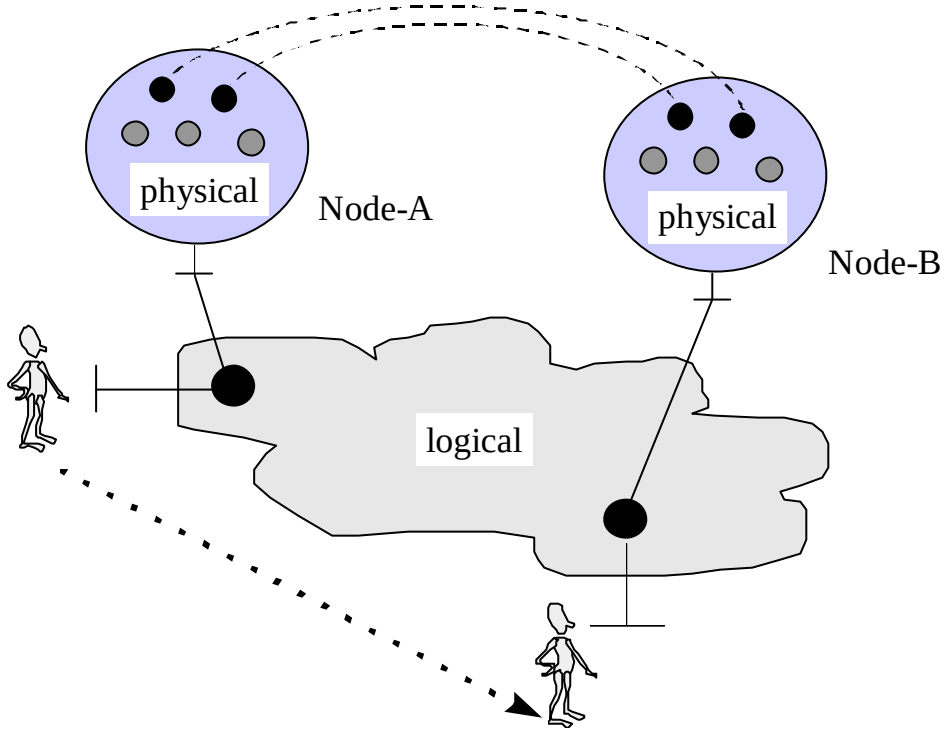


Douglas Donaldson,
APM

CEC Technical Review

10th June 1998

Information Space

Research Objectives

- MOW gives migration and relocation transparency (‘Mobile Objects’)
- InformationSpace will give persistence and failure transparency (‘Robust Objects’)
  - Persistence of the server will be transparent to both the client *and* the server
- It aims to support migration and replication of robust objects

Immediate Requirements for FollowMe

- Agents cannot be relied on to deliver results
  - Their host or network may crash
- They need a service allowing information to be stored reliably
- For example, the Newspaper Application would benefit from additional transparencies
  - Both news gathering agents and news delivery agents would gain performance benefits from transparent replication of news information objects.

Simple Black Box Model

- InformationSpace as an abstraction of a file system

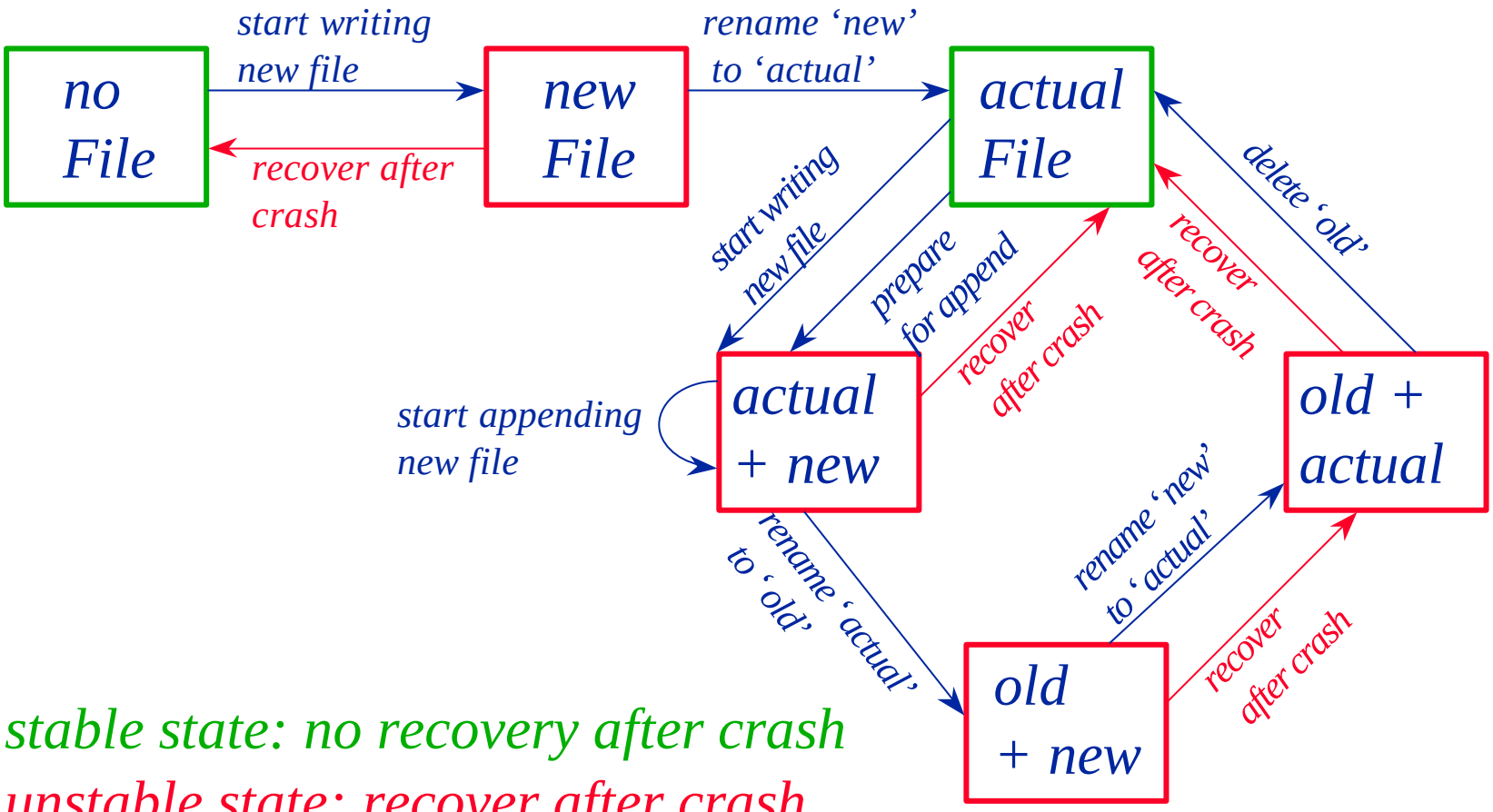
```
void copyInto(String name,
              Object obj)
Object copyOut(String name);
void remove(String name);
String[] listNames();
```

```
write(filename, buffer);


buffer = read(filename);
delete(filename);
filenames = list();
```

  - InformationSpace prevents conflicting copyIntos
  - Objects copied by value (FlexiNet model)
  - Extensible with locking, permission modes,…
  - Client manages copies, names, versioning.

State Model for Atomic Write



stable state: no recovery after crash

unstable state: recover after crash

Less Simple White Box Model

- InformationSpace as an Object Database

```
Interface newStorable(String name, Class class, Object[] args);
Interface lookup(String name);
void remove(String name);
```

  - The Storable Objects are *encapsulated* behind the IS
  - The Storable is just a (remote) object to clients
  - The Storable is (nearly) an arbitrary data type
    - so the White Box Model subsumes the Black Box Model
  - The InformationSpace prevents conflicts *and more…*

Storable Example

- ```
  public interface List {
     public void insertAtHead(Object object);
     public Object removeFromTail();
  };
  ```

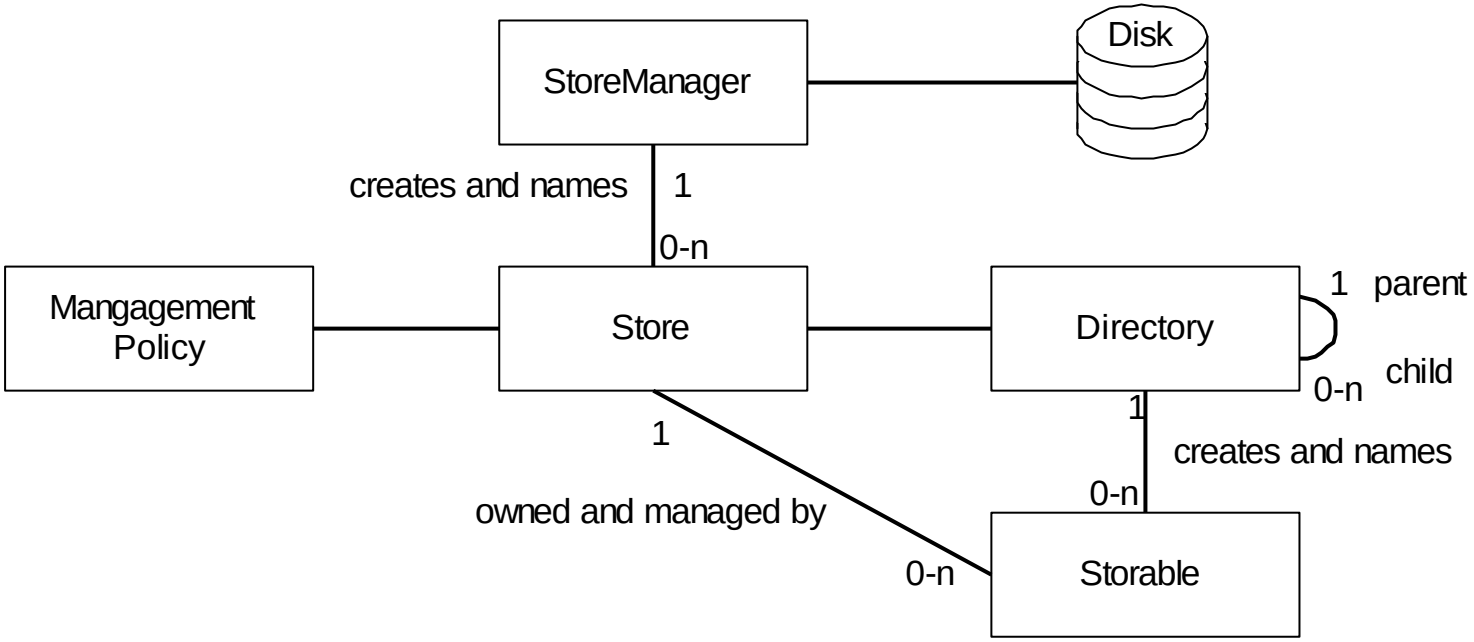
- ```
  listRef = infospace.newStorable(ListImpl.class,
                                  “liszt”);
  ```

- ```
  listRef.insertAtHead(“item”);
  ```

- The InformationSpace keeps the List persistent
  - Transparent to the client using listRef, and to List itself


  ```

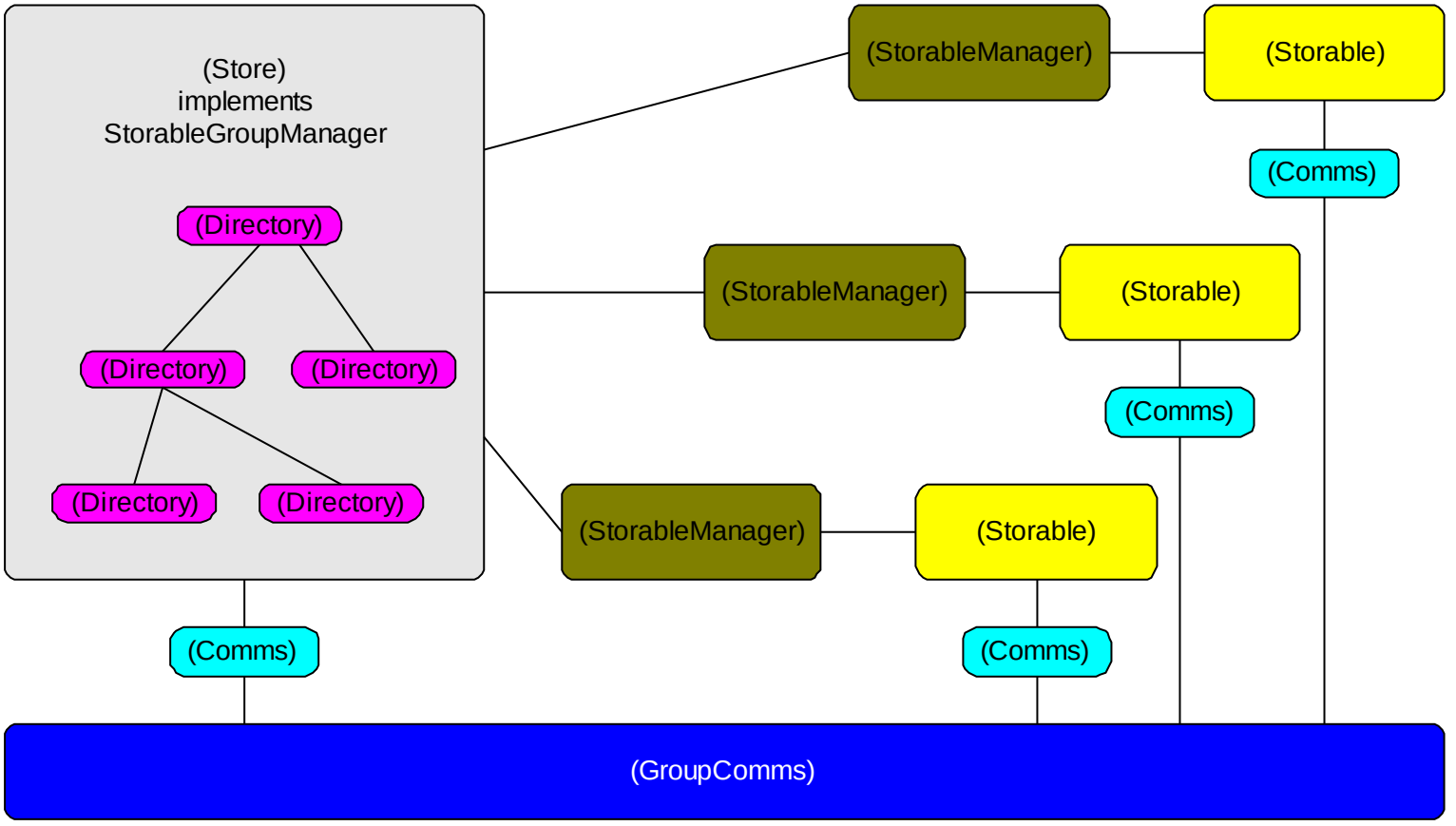
# *External View of the Design*




- The StoreManager can create Stores with different policies.
- The policy covers all the Storables in the Store

# Storables with (Group) Managers



# *MobileObjects and Storables are Clusters*

- A Cluster is a managed group of Objects
- Both Storables and MobileObjects rely on
  - Encapsulation
  - Thread management
    - The conditions for a Storable to be stored are the same as those for a MobileObject to be moved
  - Clever interface references returned on creation
- There are differences too
  - Storables are not autonomous or active

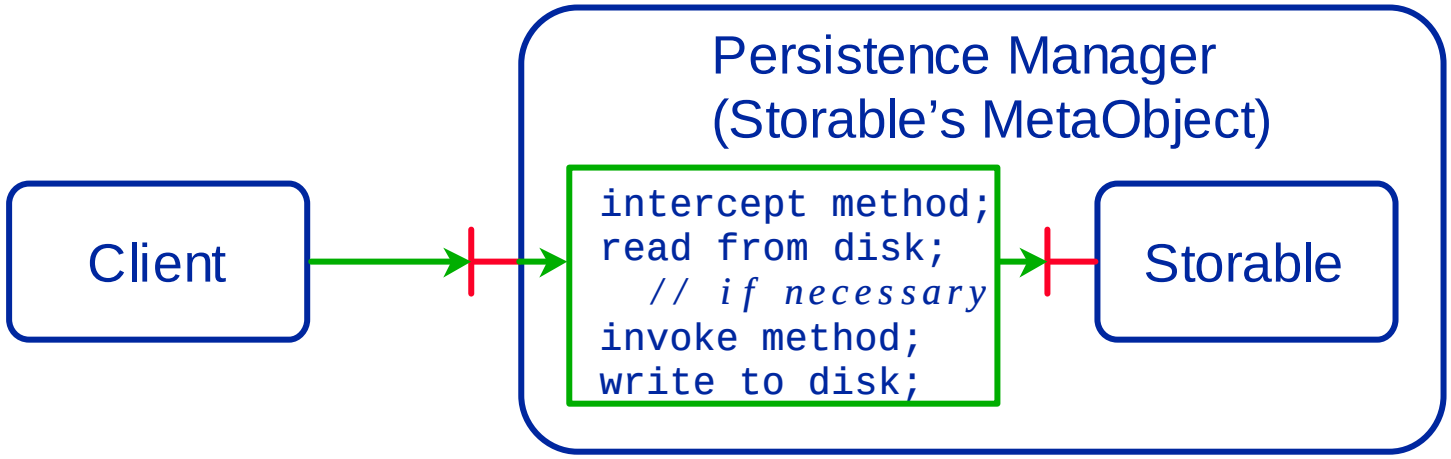
 

# *Storable(Group)Managers are MetaObjects*

- A (Group) MetaObject transparently controls Object (Group) behaviour
- A StorableManger acts as a Storables' MetaObject
  - providing persistence and failure transparency
- Aim to cleanly separate concerns of application objects from management behaviour
  - providing distribution transparencies, resource control, etc.
- Autonomous objects may manage themselves

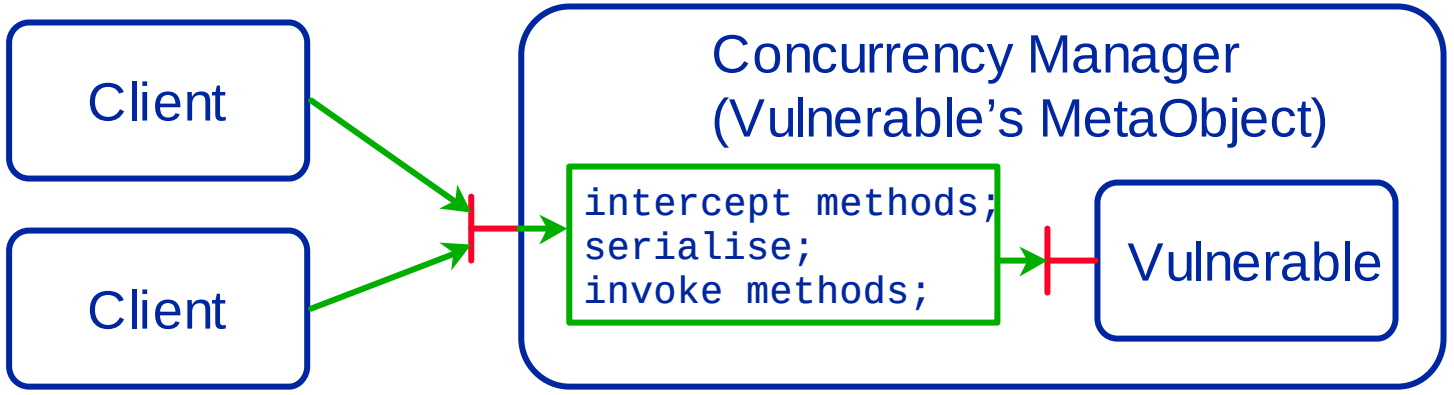
# *Managing Persistence*




- Constraints on Storable:
  - Serializable Somehow
  - Well Behaved

# *Managing Concurrency Control*




- Given the default persistence model, and unknown concurrency semantics for vulnerable, all methods must be invoked sequentially.
  - Policy for mutual exclusion could be:
    - Fair (no starvation)
    - Unfair (possible starvation)

# *Advanced Managment*

- Persistence Manager which only saves changes
- Concurrency Manager with complex scheduling
- Failure Recovery Manager which hides the recovery of failure from the client
  - Restart after failure is like restart after movement
- Transaction Manager which coordinates distributed object transactions
  - Client must have some transactional awareness
- Migration Transparency (Storable *FollowsMe*)
- Replication Transparency (Storable *FollowsThem)*

# *Summary So Far…*

- Black Box model finished, end March 1998
  - Software and Reports delivered
- White Box model in progress
  - Persistence transparency finished May 1998
  - Failure transparency underway
  - Explicit migration and replication underway
  - Research goal: transparent migration and replication