FollowMe Architecture

User Access

Richard Hayton

User Access Architecture

- We have been considering the UA Architecture
  - as part of the review process
  - in order to write the architecture document
  - because we expect to add our own UA like components
- User Access covers complex issues
  - difficult to present/understand
  - current explanation is focused on engineering
- We propose an alternative architecture
  - not suggesting a code rewrite - just a new presentation

Why?

- For review
  - easier for reviewers to understand
  - more coherent with architecture document (and other WPs)
- Leverage MOW
  - There are new MOW features that can be exploited
  - Make the FollowMe architecture more coherent
- For Modularity
  - To allow additional device types to be added without changing architecture
  - To hide implementation details from agent/application programmer

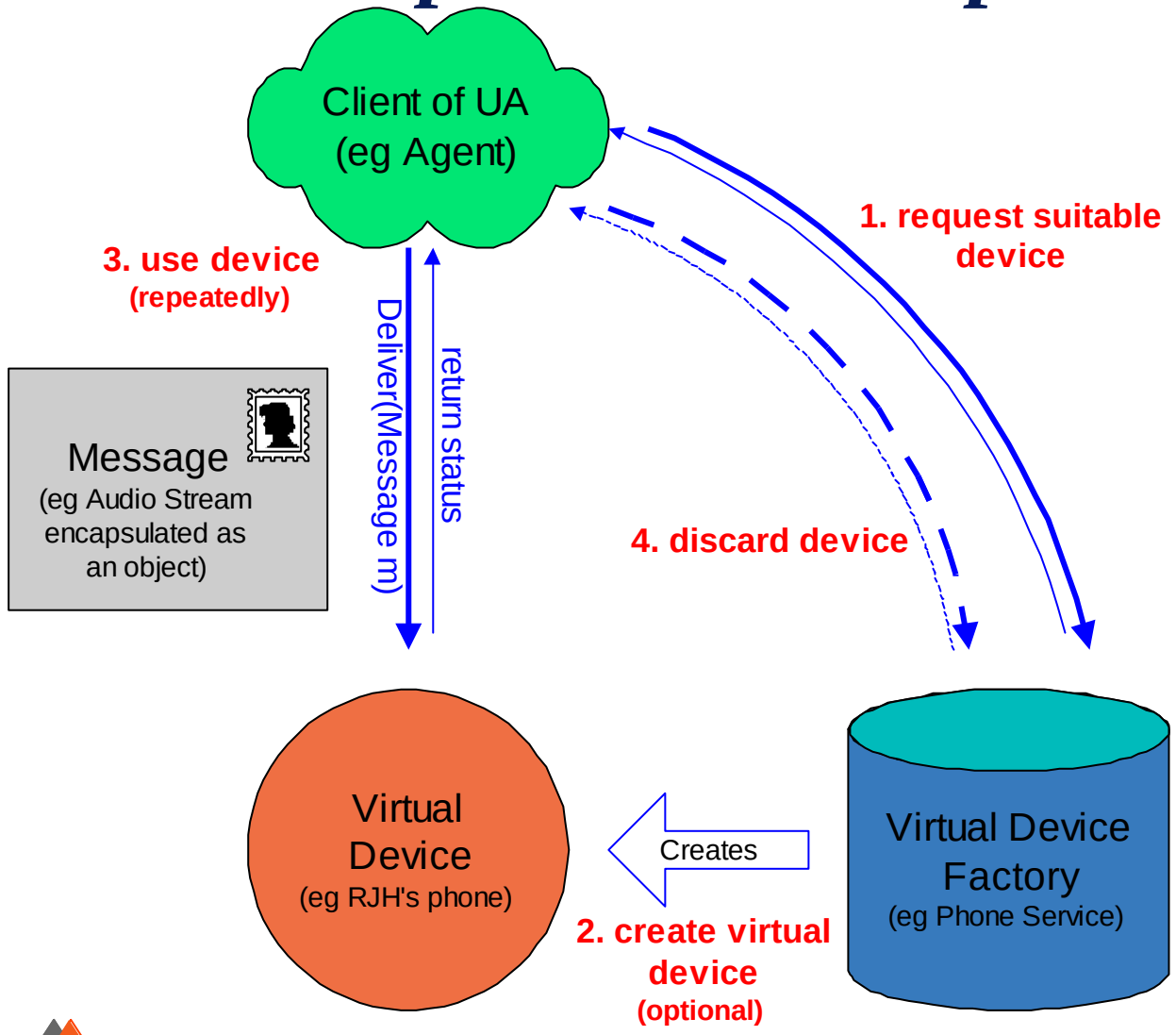
Approach

- keep it simple
  - message delivery to devices
  - hide implementation details in design
- Assume synchronous delivery
  - simpler & easier to understand
  - async is then a special case
    - same approach as MOW event service
- Special cases/extensions
  - user on line via web browsers
  - user uncontactable for long periods
  - XML / XSL

Fundamental concepts

- Messages/Documents/Deliverables
  - these are objects that are to be delivered.
- Virtual Device / Endpoint / Device Gateway
  - where a messages ends up (software object)
  - an abstraction of a particular (single) destination
    - Richard Hayton
    - Phone 01223 713111
    - 128.232.0.256:1232
- Virtual Device Factory / The User Access
  - The thing that creates/manages Virtual Devices
    - e.g. “find me Richard Hayton’s phone”
    - e.g. “connect me to Fax 01223 359779”

Relationship between components



Messages

- Messages are objects
  - There are many subclasses of message
    - e.g. XML, Voice, Text etc.
  - They have accessor methods for obtaining data
  - An implementation of a message class can use any means to service requests
    - e.g. stored state
    - format conversion
    - callback to agent (or some other services)
    - access to a Storable.
  - A message can even embody an interactive session
    - e.g. Browser receives “start applet messages”

Virtual Devices

- Virtual Devices represent endpoints
  - i.e. service + address
- They understand particular message types
  - e.g. XML, Audio….
- They may be arbitrarily complex
  - e.g. accept XML messages, remunge and then forward to another virtual device
- Are closely related to virtual device factories

Virtual Device Factories

- Manage virtual devices
- Many be simple or complex
  - e.g. federated over several machines
- May create virtual devices on demand or maintain a list
- May manage real devices
  - hidden from rest of system
- May (or may not) garbage collect virtual devices
- May (or may not) manage remote virtual devices

What about XML ?

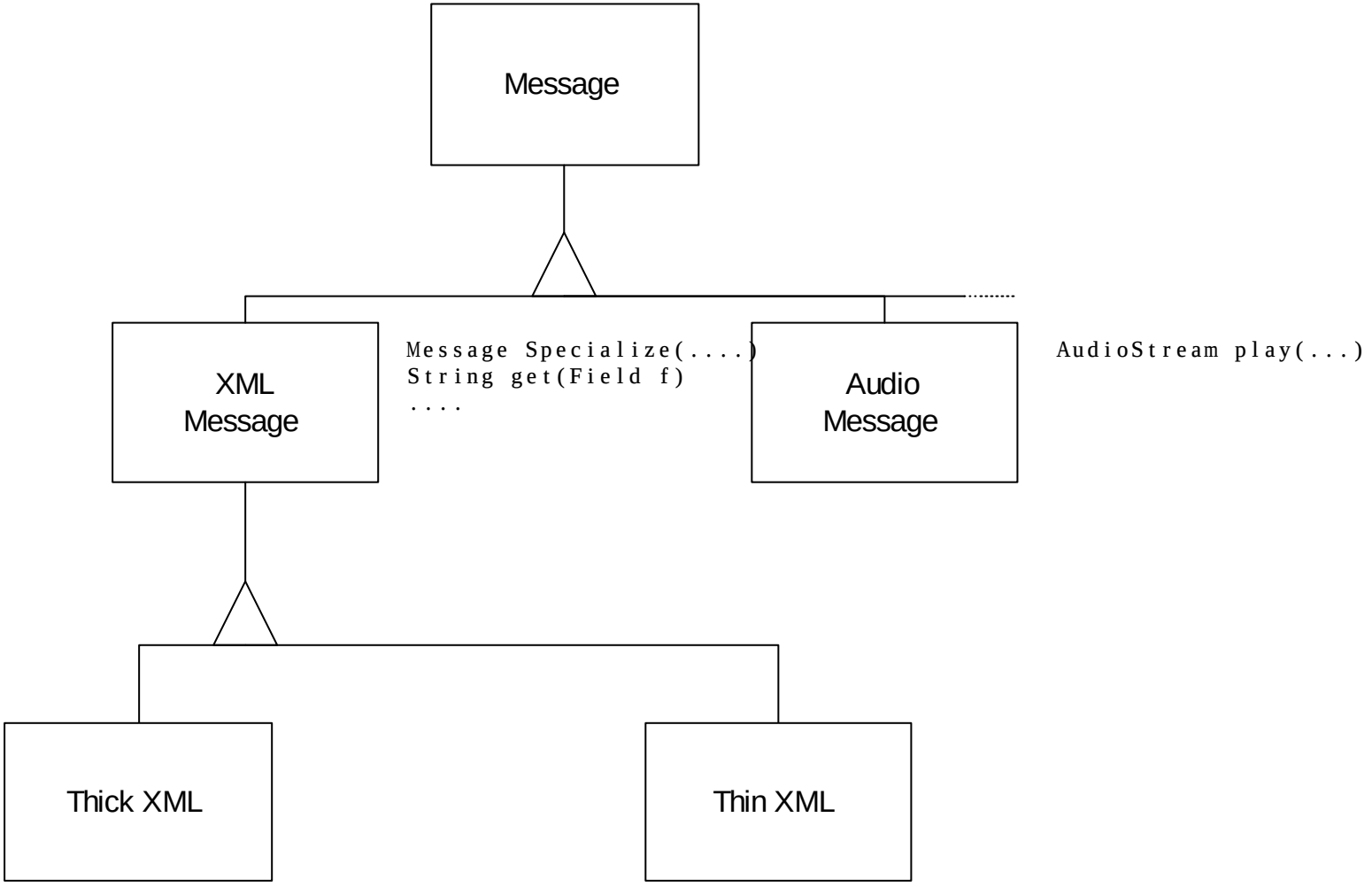
- A specialisation of the *most general case*

- Why use XML?
  - Three reasons
  - When the same message is to be sent to many clients
  - When the source and destination of the message are not available at the same time
  - To aid abstraction
    - of heterogeneous device types
    - of presentation issues for structured information

XML Implementation Choices:

- Thick Message
  - contains all the data
  - accessor functions manipulate this locally
  - needed for stored messages
- Thin Message
  - contains a reference to {a document on} a remote agent/server
  - accessor functions call back to original document
  - *encapsulation of connections/EventManagers*

Message Class Hierarchy



Relationship to current UA Design

- "The UserAccess" = A (federated) Virtual Device Factory
  - Virtual Device Factories may come in many flavours
    - e.g. There may also be non-XML versions
- Connections / Event Handlers etc. = Thin XML Message
  - We propose a changed view of encapsulation
  - connection is internal to (an) implementation of thin XML
- Logged in Users
  - next slide

Logged in users

- Provide an additional virtual device
  - This is capable of reading certain message formats
    - XML ?
    - AWT Connection Messages?
    - HTML Forms?
- Login generates an event from the “Login Service”
  - any agent/service may register interest in this
    - MOW Event Service
  - the event may contain a reference to the virtual device
- Subsumes FAST description

Effort:

- Design of Thin XML Message & XML Capable Device
  - current FAST work
- Design of other Devices
  - e.g. FaxGateway
- Design of “Logged in User” device
- Design of Virtual Device Factories
  - especially high levels ones
    - “Create device to talk to Richard on Wednesday afternoons”
    - makes use of Personal Profiles?

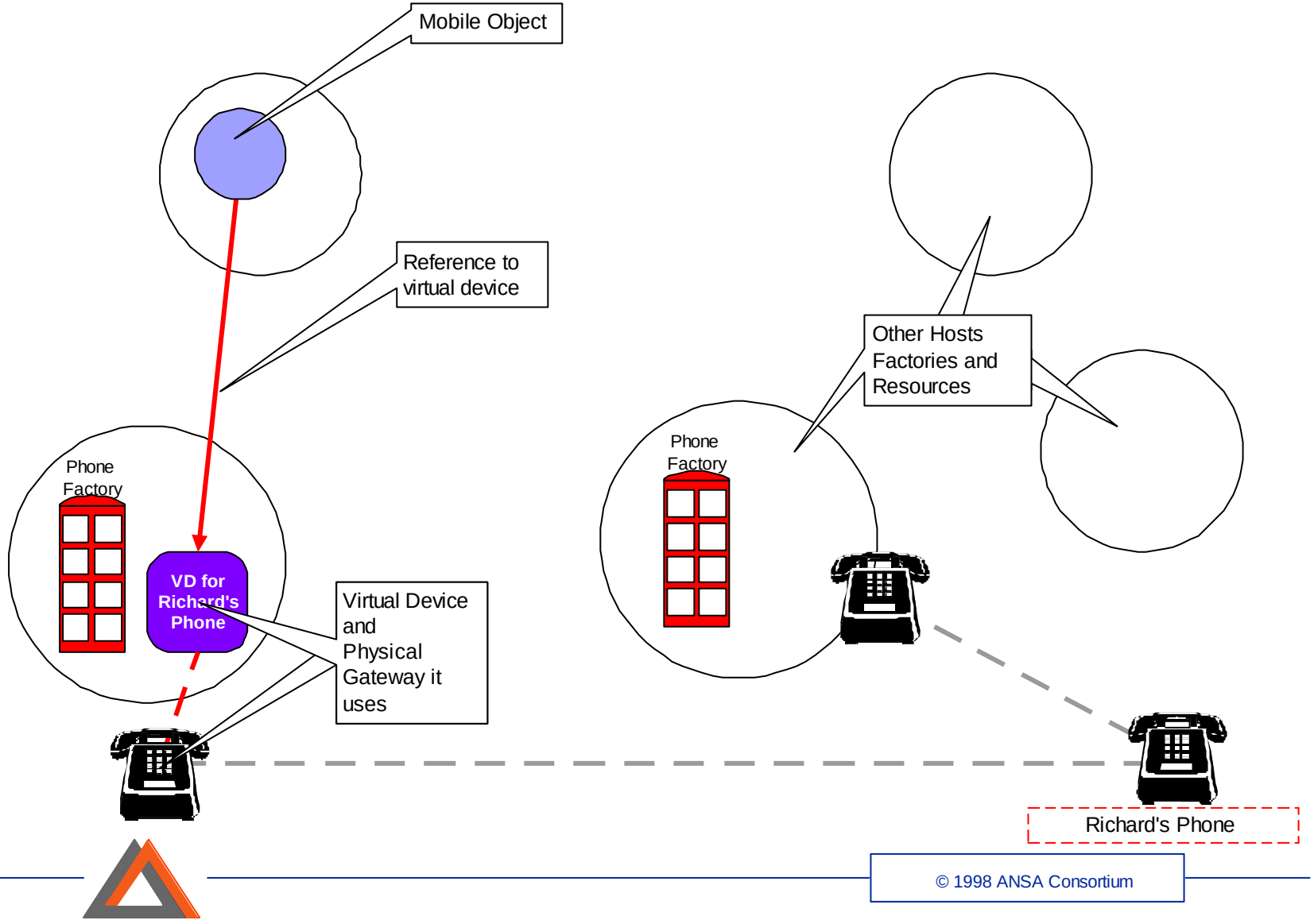
Summary

- The current User Access work contains the right bits
  - but the implementation and architecture are confused
  - as part of the architecture work package, we propose a reorganisation of the UA architecture.
- We support the use of XML
  - but the architecture ought to allow non-XML devices
  - XML is a special case
- We believe this architecture is easier to understand
  - it integrates better with other workpackages
  - is a different way of looking at the same problem & solution

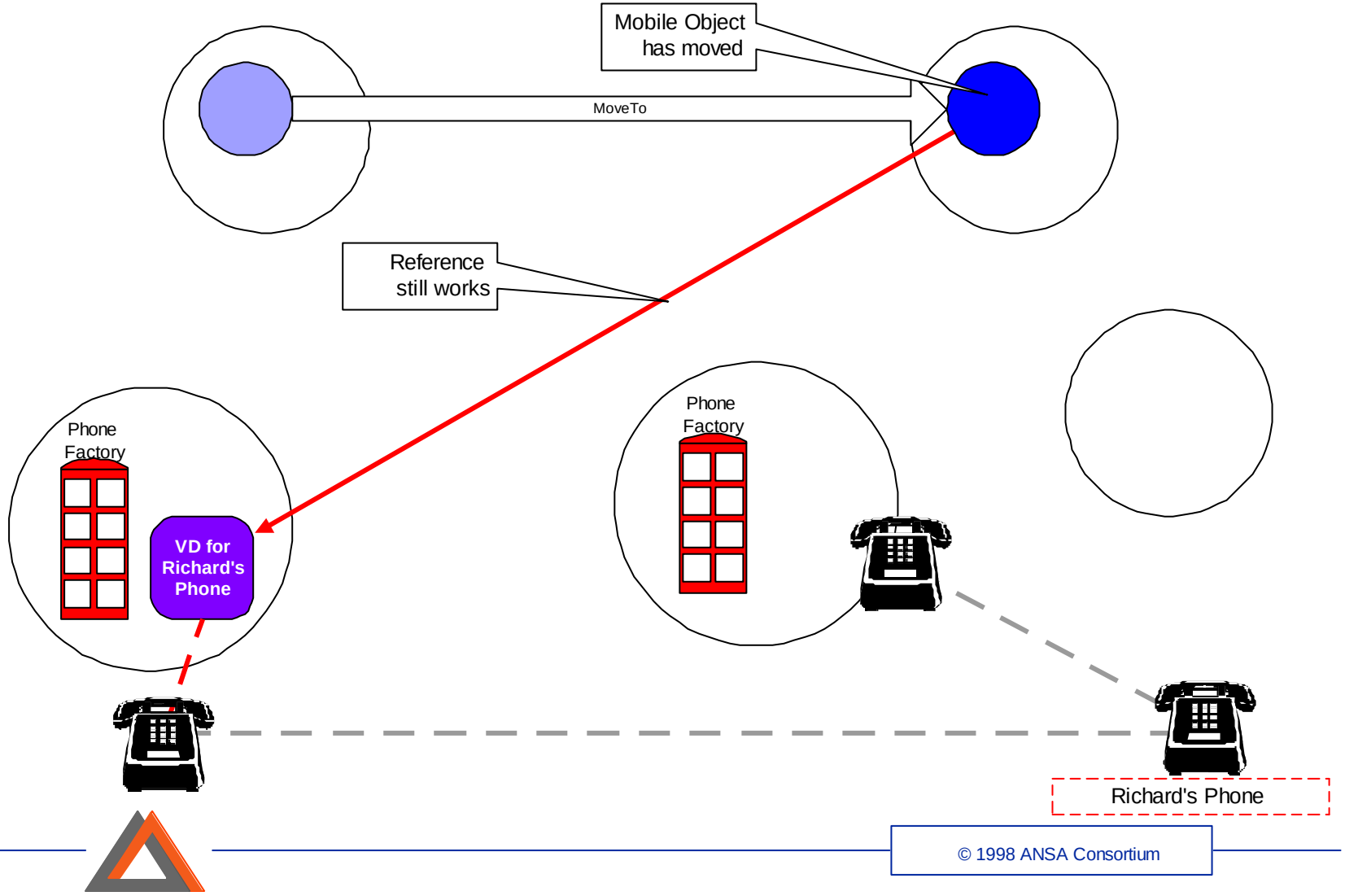
UA as a set of MOW Objects

- References to virtual devices may be stored, copied etc.
  - consistent architecture
- When an agent moves, it may keep a reference to the virtual device.
  - The MOW will maintain this reference
  - It may be transparently *improved*
    - This issue applies equally to other objects (e.g. storable)
    - It is a MOW Architecture issue

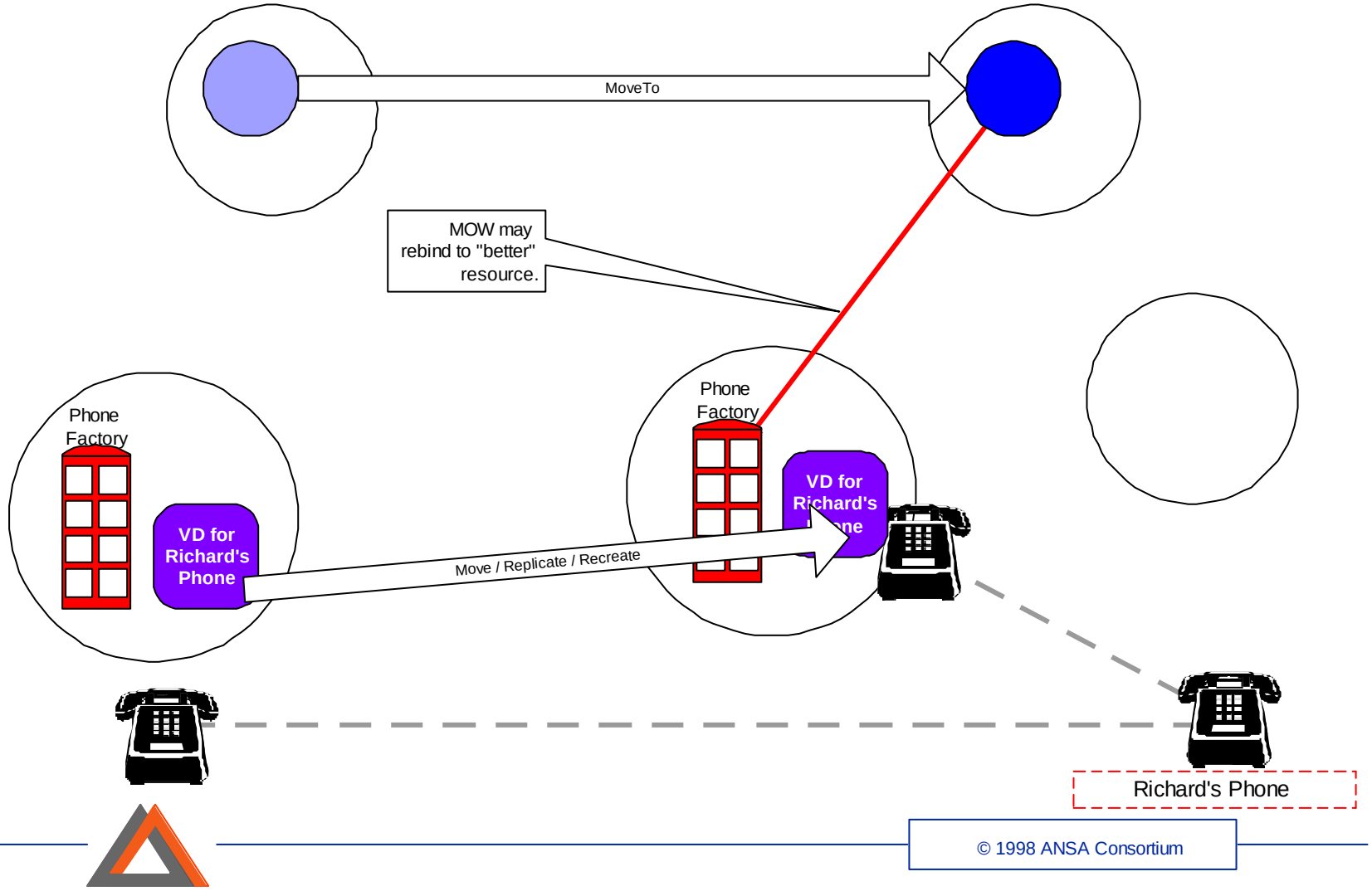
Mobile Object using Virtual Device



After Migration all is well



Transparent Optimisation by MOW



Summary

- The current User Access work contains the right bits
  - but the implementation and architecture are confused
  - as part of the architecture work package, we propose a reorganisation of the UA architecture.
- We support the use of XML
  - but the architecture ought to allow non-XML devices
  - XML is a special case
- We believe this architecture is easier to understand
  - it integrates better with other workpackages
  - is a different way of looking at the same problem & solution