Sistemi Distribuiti
Corso di Laurea in Ingegneria

Dr. Davide Rogai, Prof. Paolo Nesi

PARTE 7: **SW Components, and M3W**

Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796469

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet

nesi@dsi.unifi.it paolo.nesi@unifi.it
www: http://www.dsi.unifi.it/~nesi

Why Components?

What is the motive for producing, distributing, buying, or using software components?

What are the benefits of component software?

The simplest answer is:

Components are the way to go because all other engineering disciplines introduced components as they become mature - and still use them.

Szyperski 1999

Components

- Components provide a service without regard to where the component is executing or its programming language
  - A component is an independent executable entity that can be made up of one or more executable objects;
  - The component interface is published and all interactions are through the published interface;

Component definitions

- Councill and Heinmann:
  - *A software component is a software element that conforms to a component model and can be independently deployed and composed without modification according to a composition standard.*
- Szyperski:
  - *A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties.*

Component as a service provider

- The component is an independent, executable entity. It does not have to be compiled before it is used with other components.
- The services offered by a component are made available through an interface and all component interactions take place through that interface.

Component characteristics 1

Standardised	Component standardisation means that a component that is used in a CBSE process **has to conform to some standardised component model**. This model may define component interfaces, component meta-data, documentation, composition and deployment.
Independent	A component should be independent – it should be possible to compose and **deploy it without having to use other specific components**. In situations where the component needs externally provided services, these should be explicitly set out in a 'requires' interface specification.
Composable	For a component to be composable, all external **interactions must take place through publicly defined interfaces**. In addition, it must provide external access to information about itself such as its methods and attributes.

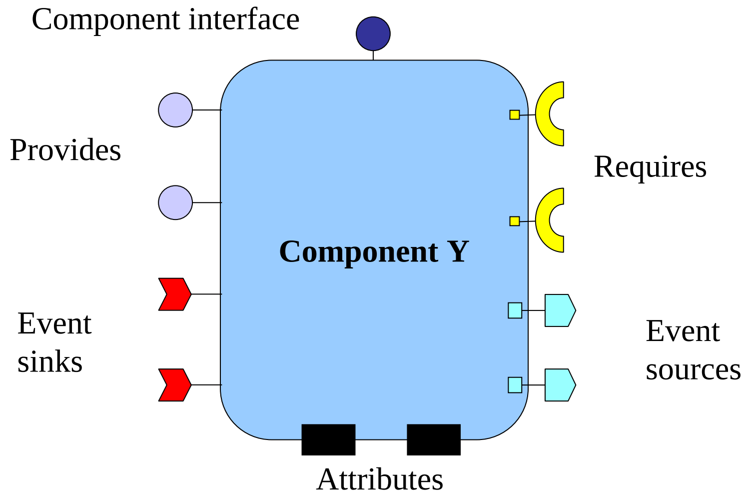
Component characteristics 2

Deployable	To be deployable, a component **has to be self-contained** and must be able **to operate as a stand-alone entity** on some component platform that implements the component model. This usually means that the component is a binary component that **does not have to be compiled before it is deployed.**
Documented	Components have to be fully documented so that potential users of the component **can decide whether or not they meet their needs**. The syntax and, ideally, the semantics of all component interfaces have to be specified.

Component interfaces

- Provides interface
  - Defines the services that are provided by the component to other components.
- Requires interface
  - Defines the services that specifies what services must be made available for the component to execute as specified.

Component Model… more complex



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 9

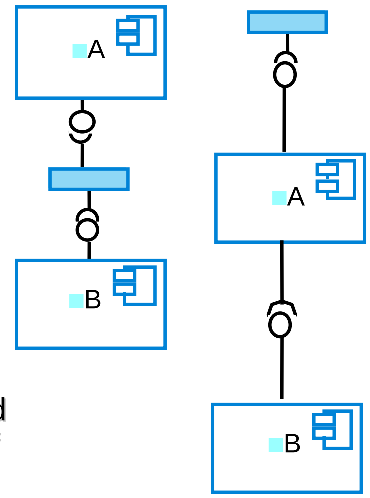
Component composition

- The process of assembling components to create a system.
- Composition involves integrating components with each other and with the component infrastructure.
- Normally you have to write 'glue code' to integrate components.

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 10

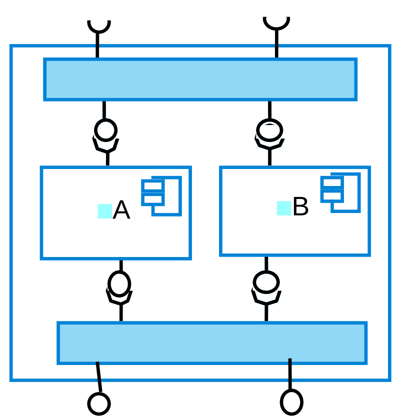
Types of composition

- Sequential composition
  where the composed components are executed in sequence. This involves composing the provides interfaces of each component.
- Hierarchical composition
  where one component calls on the services of another. The provides interface of one component is composed with the requires interface of another.



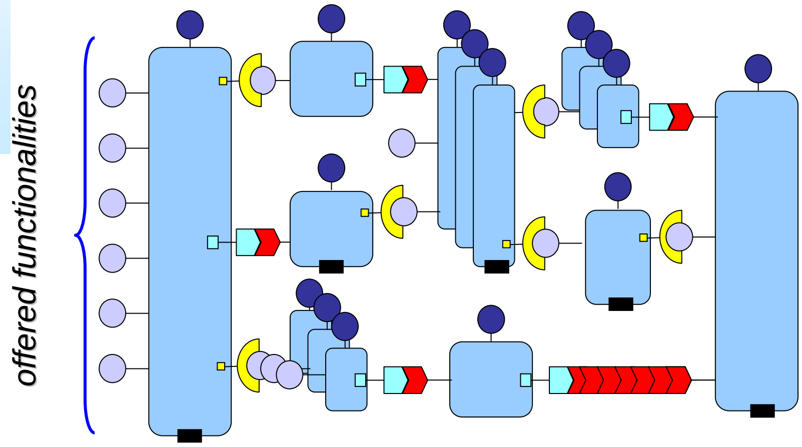
Types of composition

- Additive composition
  where the interfaces of two components are put together to create a new component.

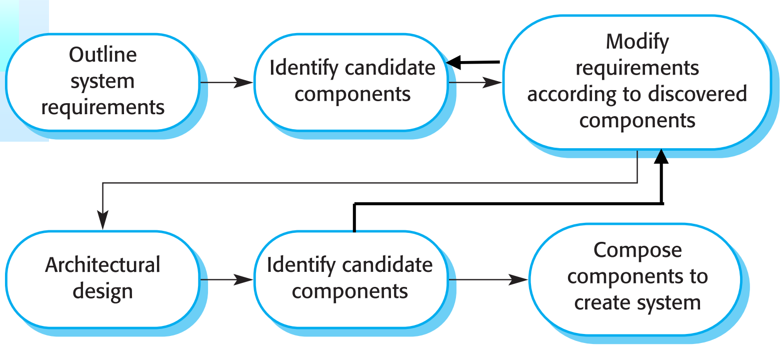


What is intended for composition

- To build a new Component by using existing ones



The CBSE process



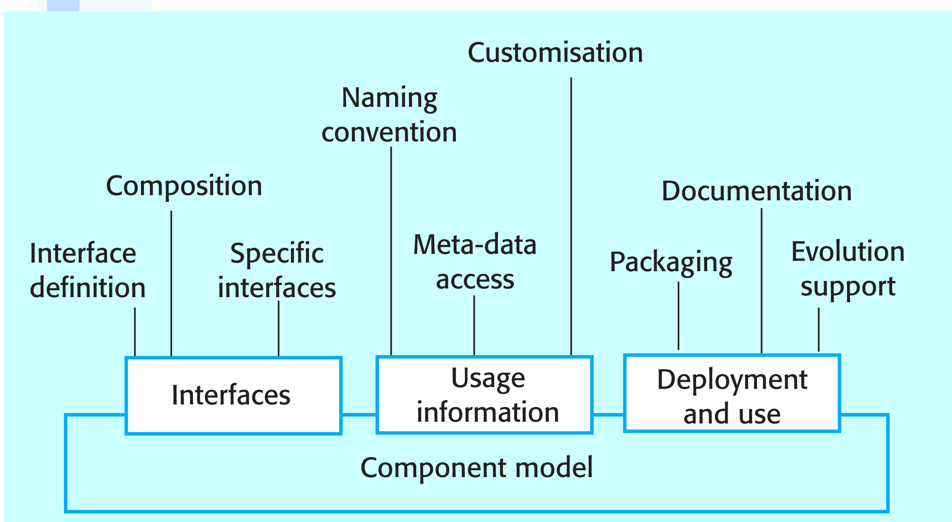
From a presentation by Ian Sommerville

Component models

- A component model is a definition of standards for component implementation, documentation and deployment.
- Examples of component models
  - EJB model (Enterprise Java Beans)
  - COM+ model (.NET model)
  - Corba Component Model
- The component model specifies how interfaces should be defined and the elements that should be included in an interface definition.

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 **15**

Elements of a component model



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 **16**

Middleware support

- Component models are the basis for middleware that provides support for executing components.
- Component model implementations provide:
  - Platform services that allow components written according to the model to communicate;
  - Horizontal services that are application-independent services used by different components.
- To use services provided by a model, components are deployed in a container. This is a set of interfaces used to access the service implementations.

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 17

M3W

ISO/IEC JTC1/SC29 WG11

MPEG

MOVING PICTURE EXPERTS GROUP

MPEG MultiMedia Middleware

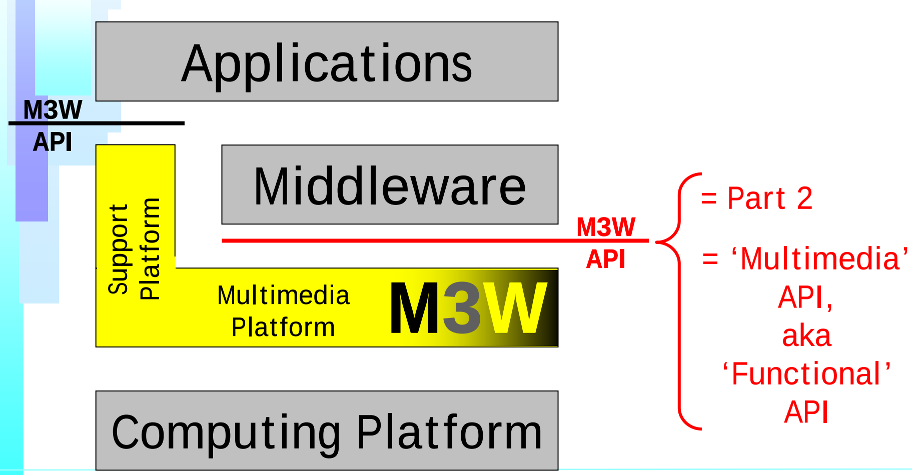
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 18

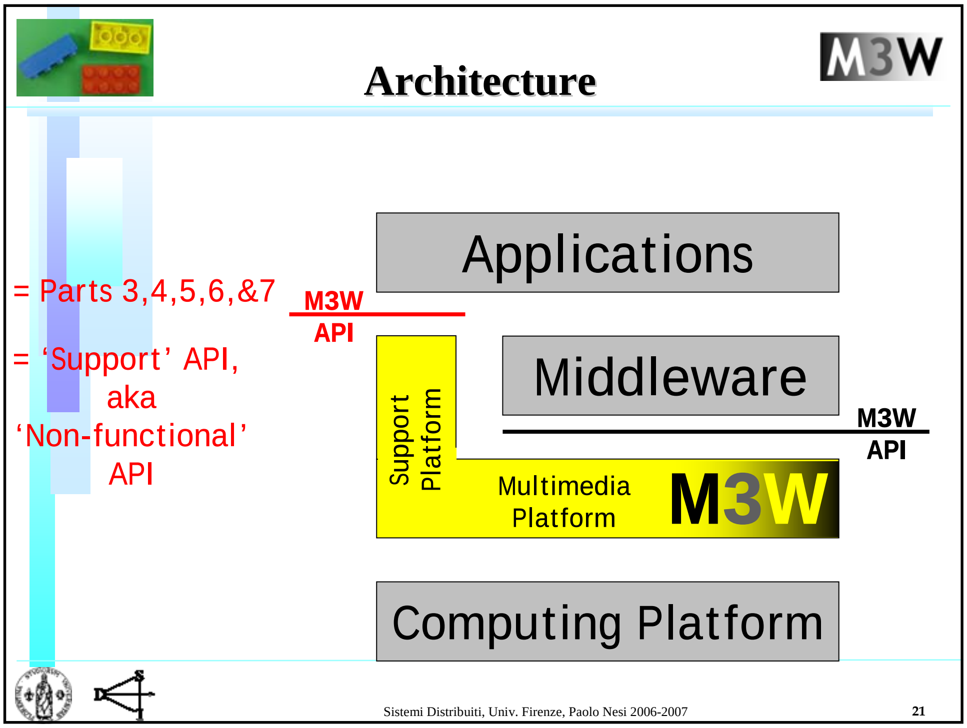
M3W is split into several parts.

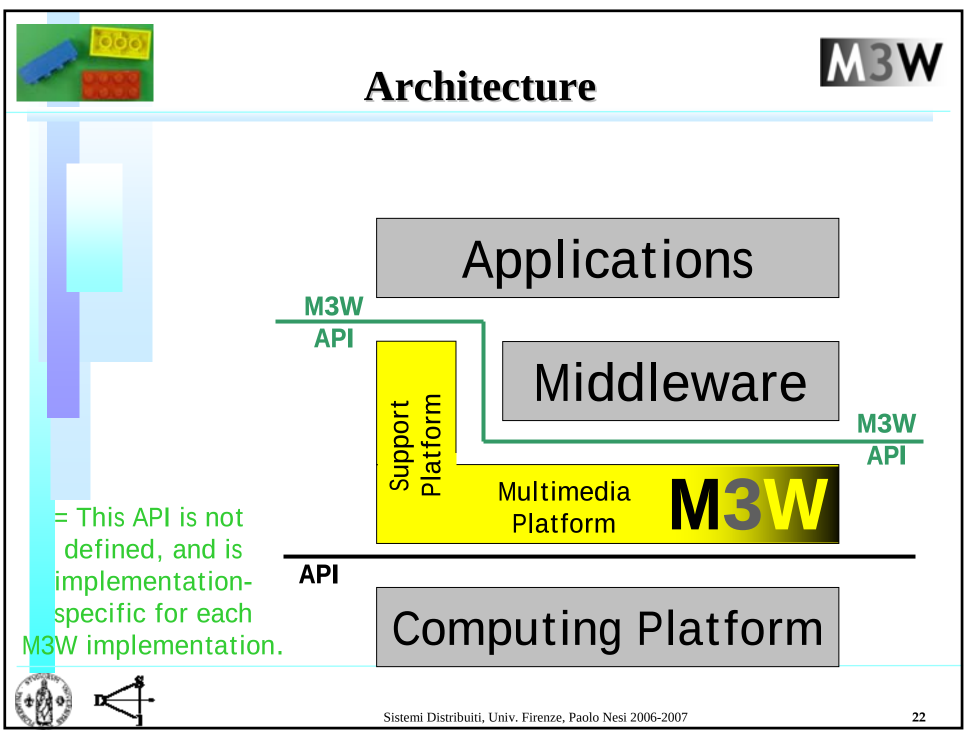
M3W Parts:

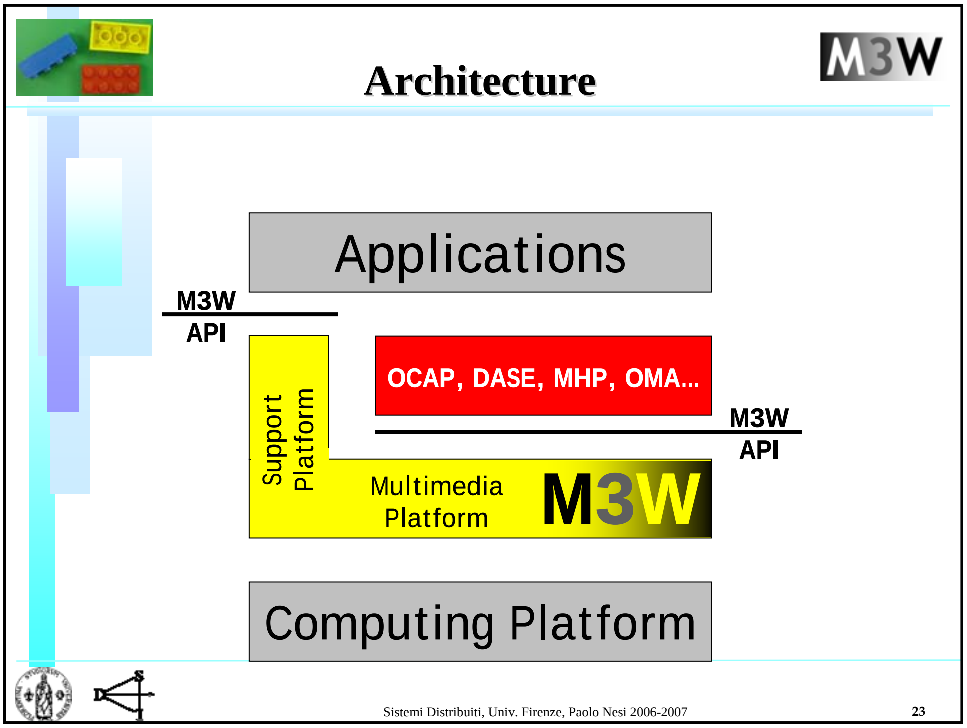
1. Architecture
2. Multimedia API
3. Component Model
4. Resource Management
5. Download / Delivery
6. Fault Management
7. Integrity Management

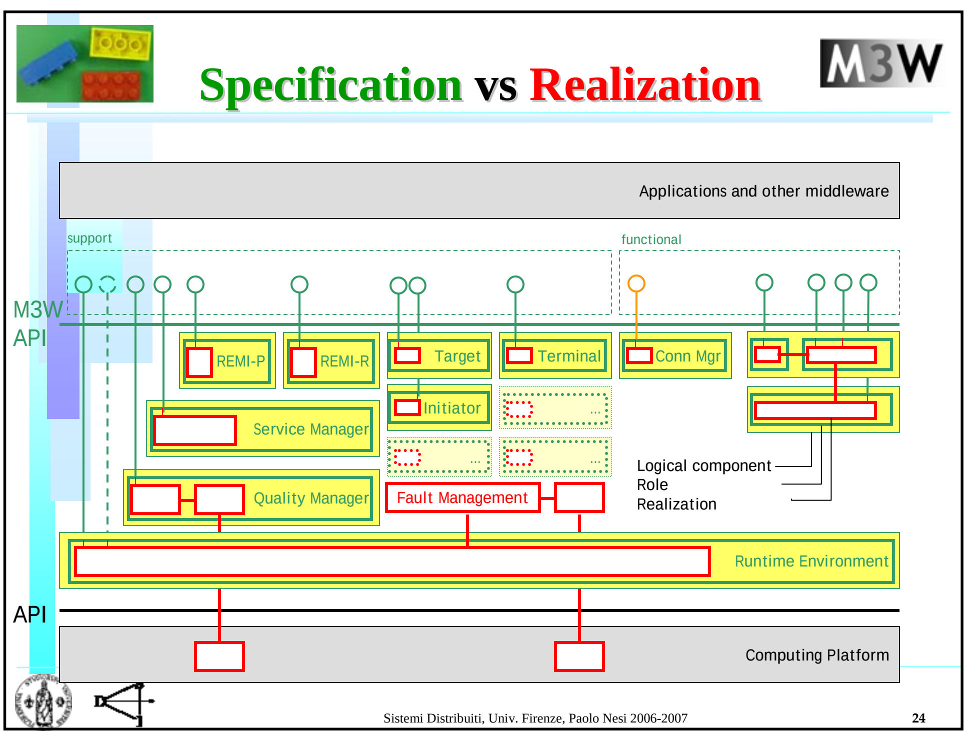
Architecture

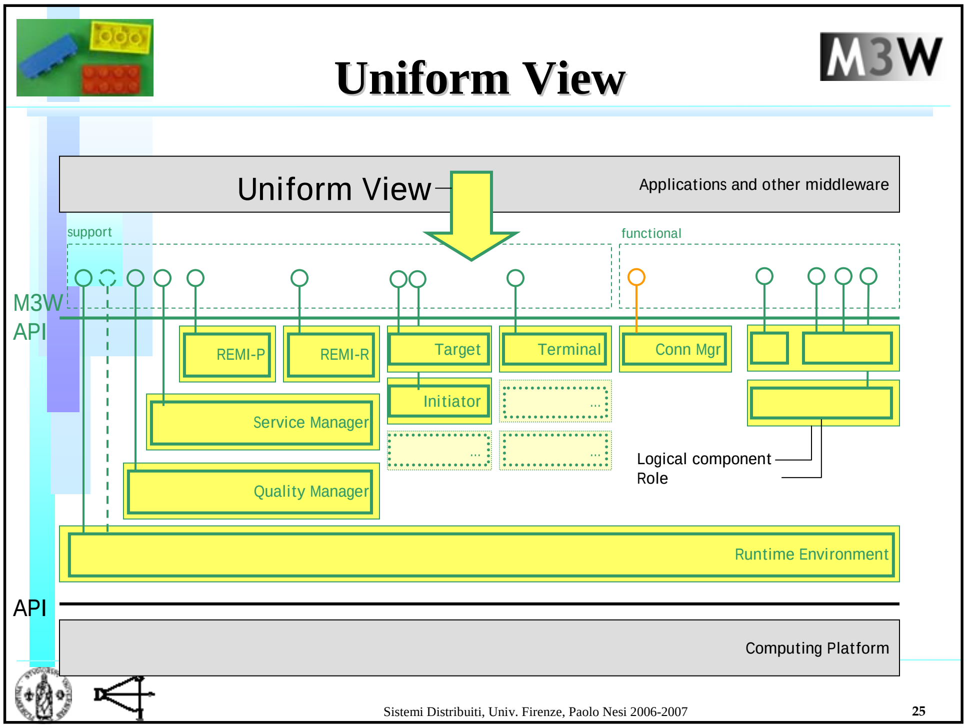


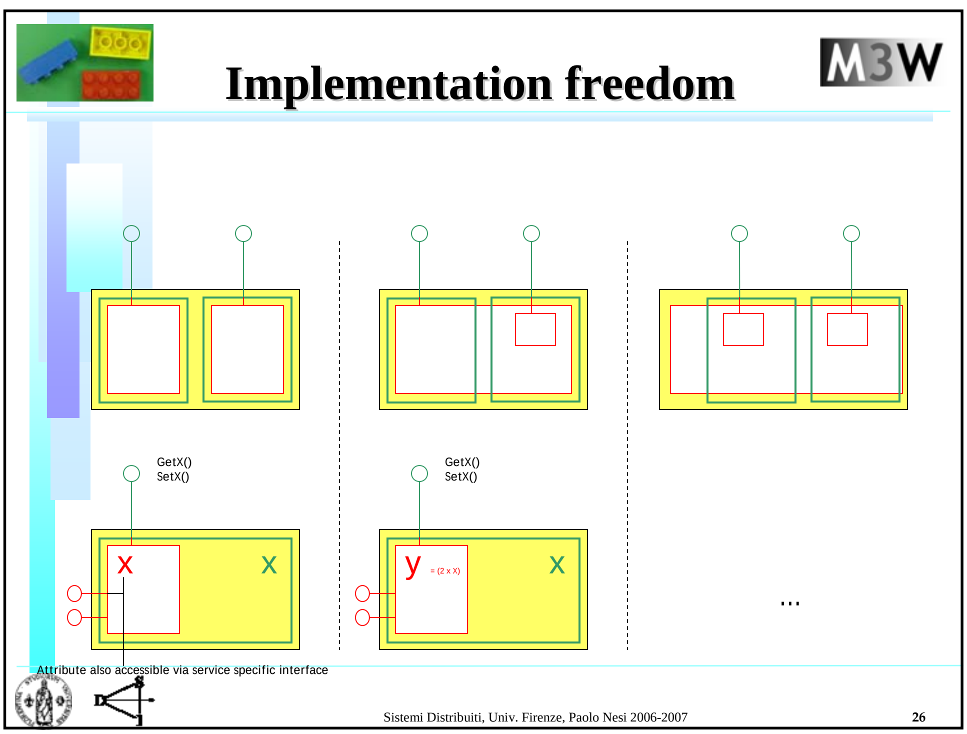

Architecture
M3W
= Parts 3,4,5,6,&7
= 'Support' API,
aka
'Non-functional'
API
M3W
API
Applications
Support
Platform
Middleware
M3W
API
Multimedia
Platform
M3W
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
21

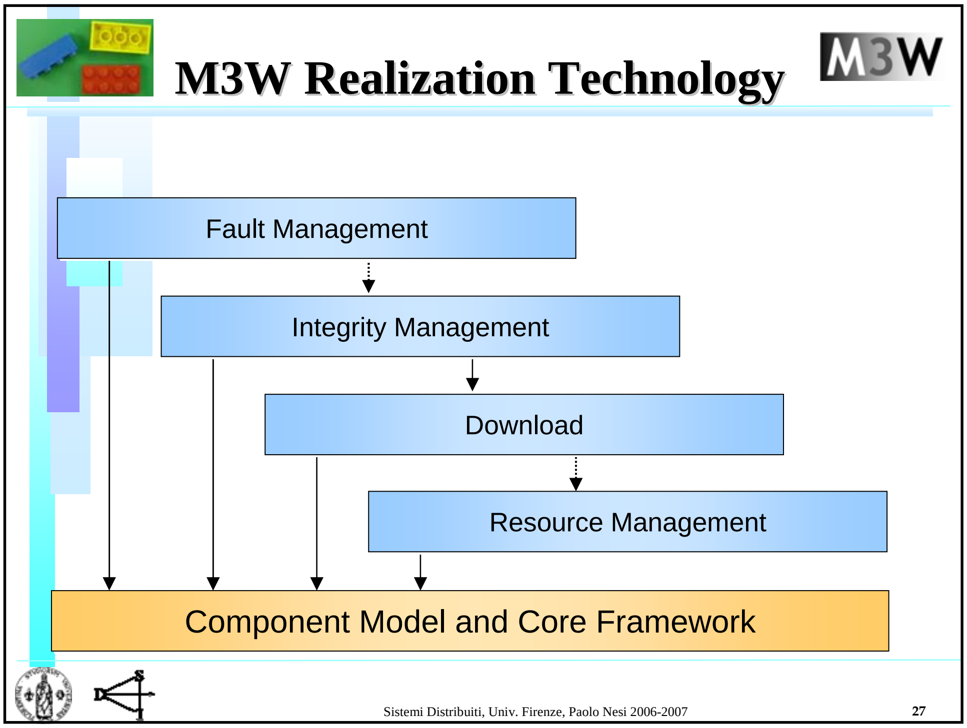

Architecture
M3W
M3W
API
Applications
Support
Platform
Middleware
M3W
API
Multimedia
Platform
M3W
= This API is not
defined, and is
implementation-
specific for each
M3W implementation.
API
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
22

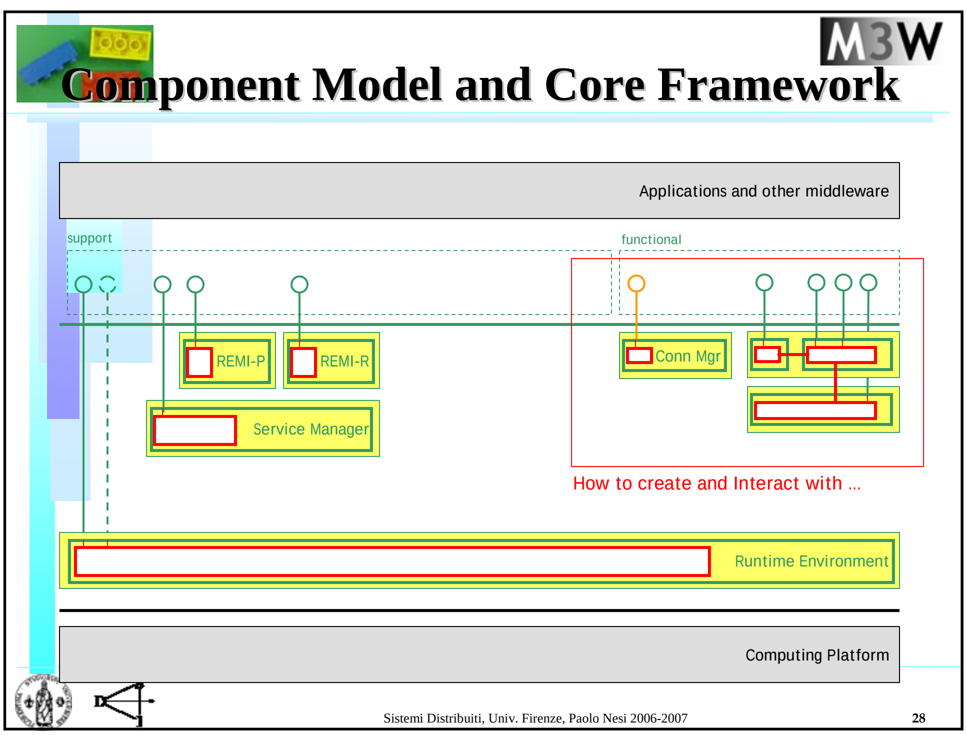

Architecture
M3W
Applications
M3W
API
Support
Platform
OCAP, DASE, MHP, OMA...
M3W
API
Multimedia
Platform
M3W
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
23


Specification vs Realization
M3W
Applications and other middleware
support
functional
M3W
API
REMI-P
REMI-R
Target
Terminal
Conn Mgr
Initiator
Service Manager
Quality Manager
Fault Management
Logical component
Role
Realization
Runtime Environment
API
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
24

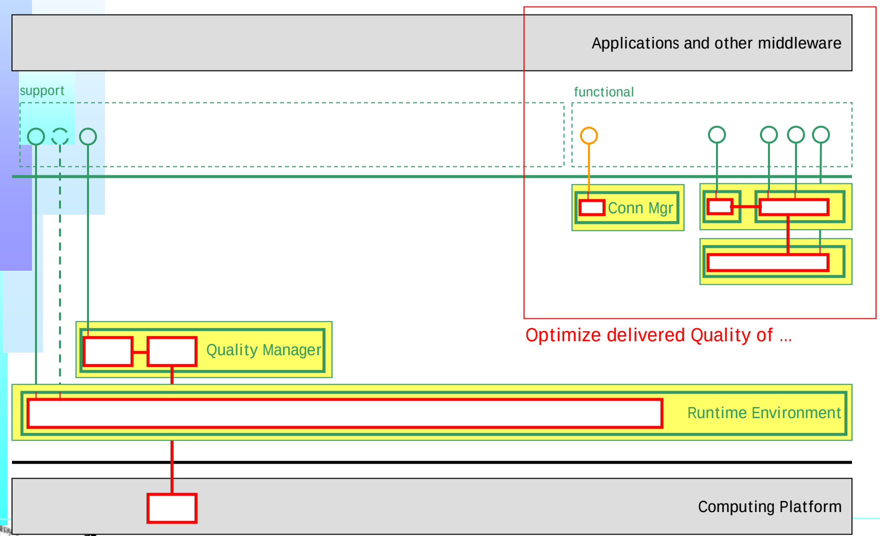

Uniform View
M3W
Uniform View
Applications and other middleware
support
functional
M3W API
REMI-P
REMI-R
Target
Terminal
Conn Mgr
Initiator
Service Manager
Quality Manager
Logical component
Role
Runtime Environment
API
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
25


Implementation freedom
M3W
GetX()
SetX()
GetX()
SetX()
x
X
y
X
...
Attribute also accessible via service specific interface
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
26


M3W Realization Technology
M3W
Fault Management
Integrity Management
Download
Resource Management
Component Model and Core Framework
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
27

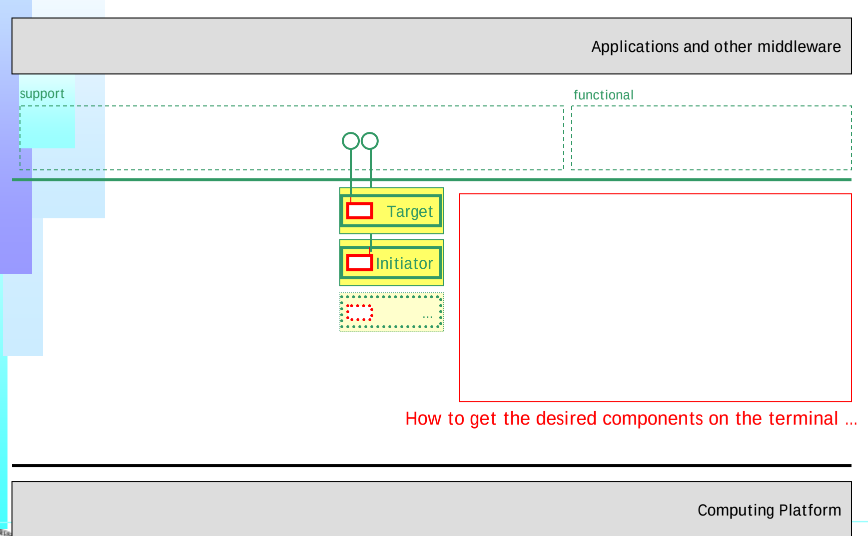

Component Model and Core Framework
M3W
Applications and other middleware
support
functional
REMI-P
REMI-R
Service Manager
Conn Mgr
How to create and Interact with …
Runtime Environment
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
28

Resource Management



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 29

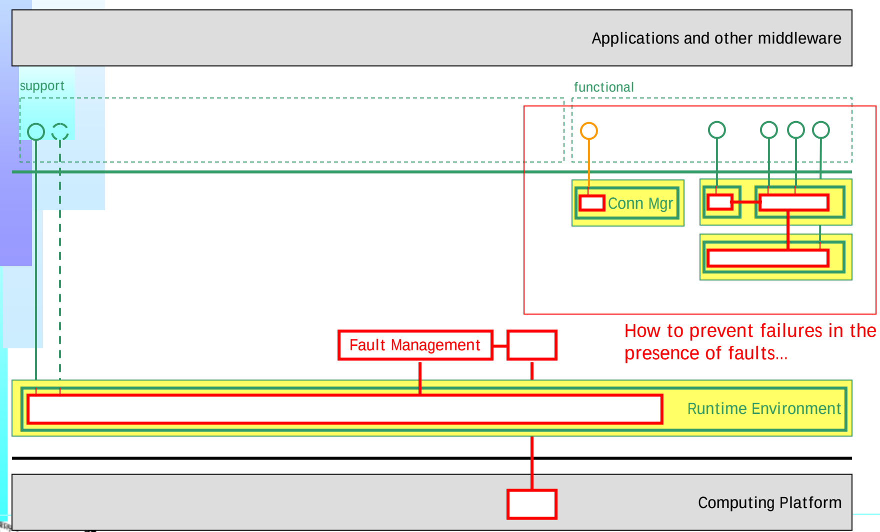
Download



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 30

Fault Management

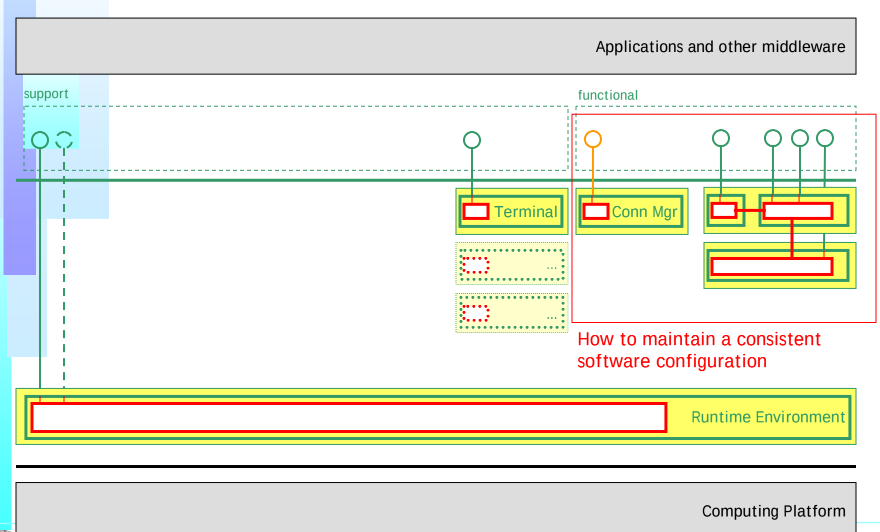
M3W



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 31

Integrity Management

M3W



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 32

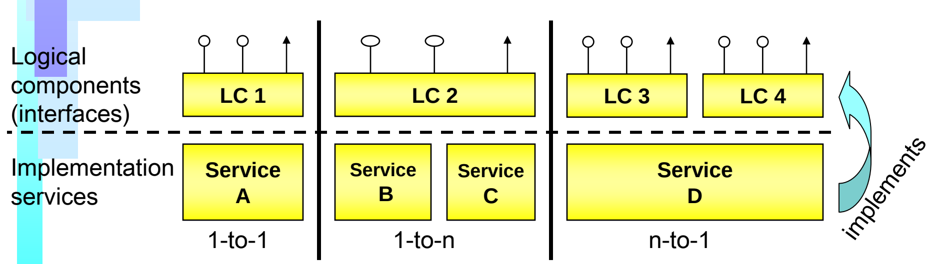
MM API does not define implementation

- Logical Component = API specification
- Implementation services need to implement logical components
- Implementer is free is choose the architecture of the implementation services
  - Implementation can for instance use hardware streaming, software streaming or a combination
  - Client does not need to be aware of implementation services

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 33

Logical services and implementations

- Implementation services do not need to map 1-to-1 to logical components



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 34

Application portability: third-party binding

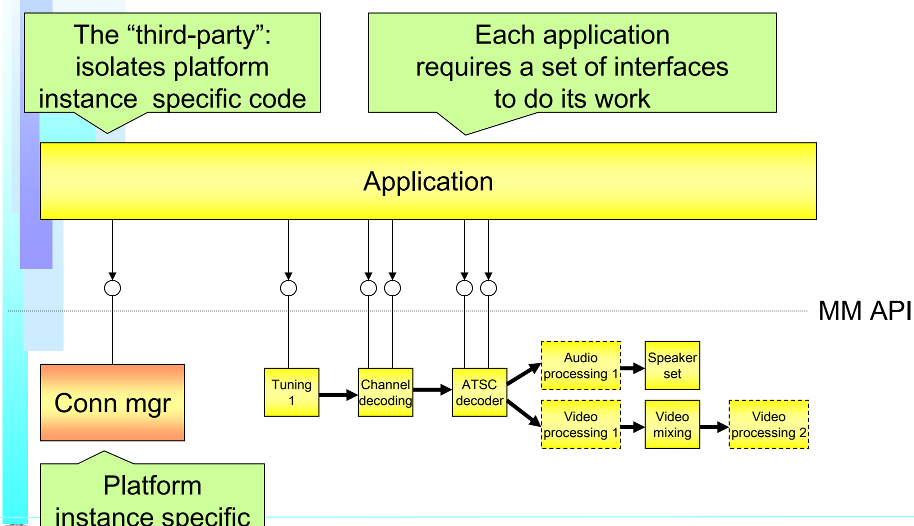
- Applications should only depend on the logical components they control
- This makes them portable across a range of platform instances
- Platform instance specific code should be isolated

Third-party binding is a key technique to enable this

The use of third-party binding is recommended, not mandatory

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 35

Third-party binding: “The players”



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 36

Multimedia API

- Audio and Video API
  - Front-end components
  - Encoders / Decoders
  - Video processing
  - Audio processing
  - Generic
- IPMP API

} Logical Components

Audio Video Logical Components

M3W

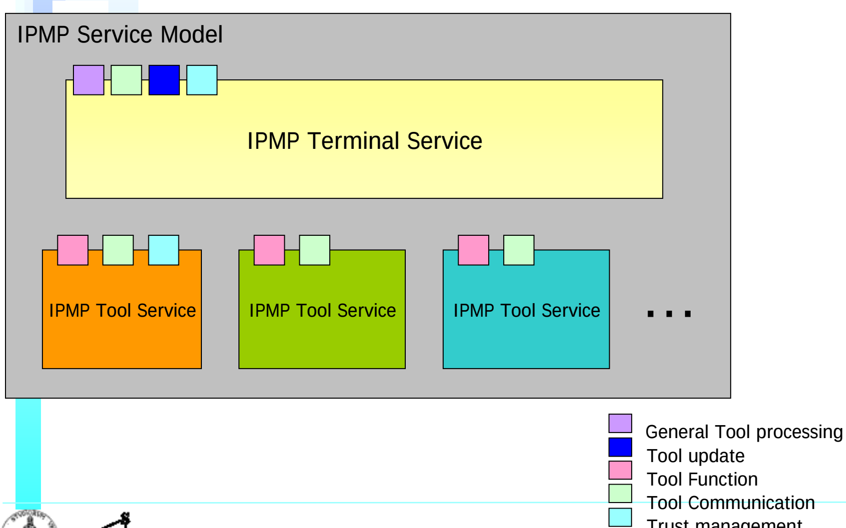
(Version 1.0)

- Front-end component (11)
  - Channel Decoding
  - Tuning
  - HdmiIn
  - …
- Decoders/encoders (5)
  - ATSC Decoder
  - Transport Stream Demultiplexing
  - …
- Video processing (16)
  - Basic Video Featuring
  - Color Transient Improvement
  - Sharpness Measurement
  - Video Mixing
  - ….
- Audio processing (10)
  - Audio Bass Enhancements
  - Audio Dynamic Range Control
  - Audio Volume Control
  - …
- Generic (8)
  - Connection Management
  - Fatal Error Handling
  - Unknown
  - …

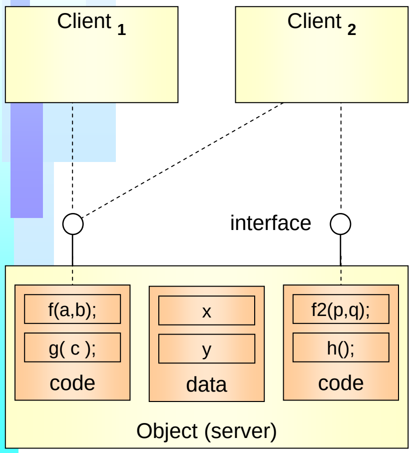
Multimedia API

- Audio and Video API
- IPMP API
  - Trust Management Interfaces
    - Key Management
    - Signature Management
    - Licence Management
    - Certificate Management
  - Tool Interfaces
    - General Tool processing
    - Tool Function
    - Tool Update
    - Tool Communication

IPMP Service Model



Objects & Interfaces



The M3W programming model is based on the *Object Oriented* paradigm

Objects are opaque representations of functionality (i.e. purely functional)

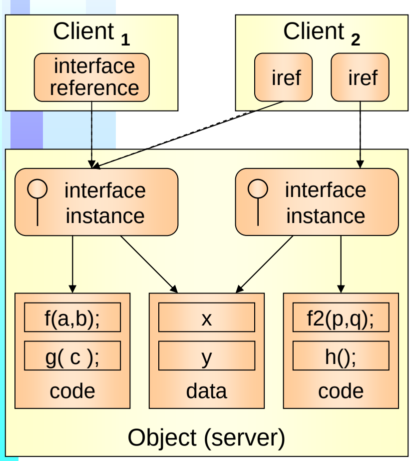
- Behavior (code)
- State (data)

The functionality is accessed through 1 or more *interfaces*

- Sequence of parameterized operations
- The object "is-a" of the interface type

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 41

Objects & Interfaces (2)



At run-time, an *interface instance* represents an interface implementation

- Is typed by the interface
- Refers to the code & data

An *Interface Reference* (*iref*) is the client's access point to the interface instance

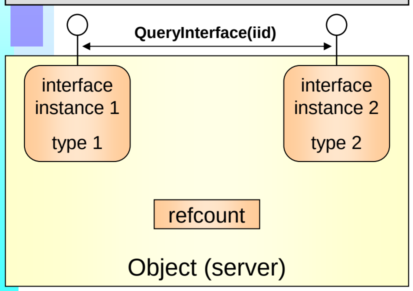
- Multiple irefs may refer to the same interface instance

Note: this is classical Microsoft COM ☺

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 42

Objects & Interfaces (3)

```
interface rcIUnknown { UUID }
{
  native QueryInterface(iid);
  long AddRef();
  long Release();
};
```



Interface definitions may *inherit* from other interface definitions

- Extend function → *subtype*

All interface definitions implicitly inherit from rc**IUnknown**

An object that implements multiple Interface *types* is *polymorph*

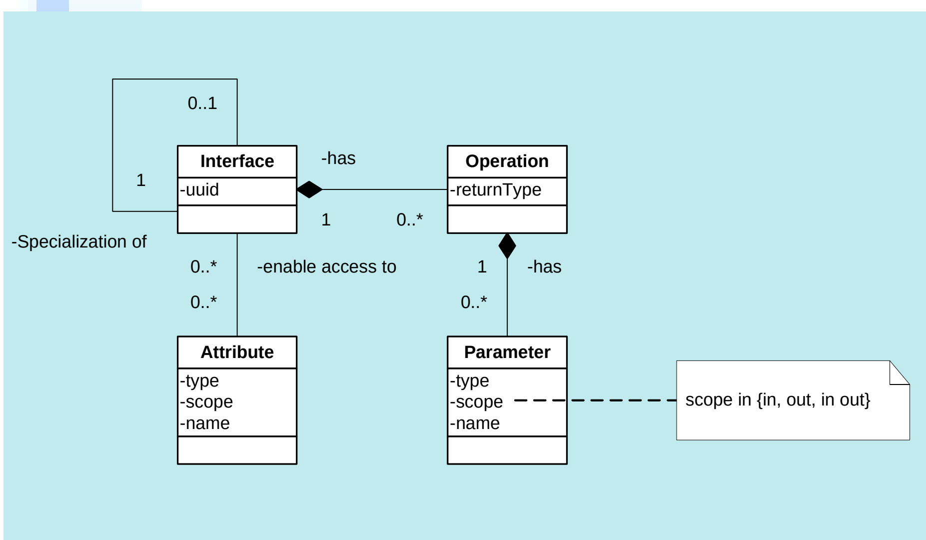
- Navigation **queryInterface(iid)**
- Dynamic discovery → flexibility

Distributed co-operative lifetime mgt

- **AddRef()**
- **Release()**

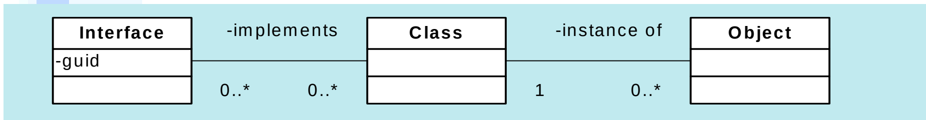
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 43

Interface



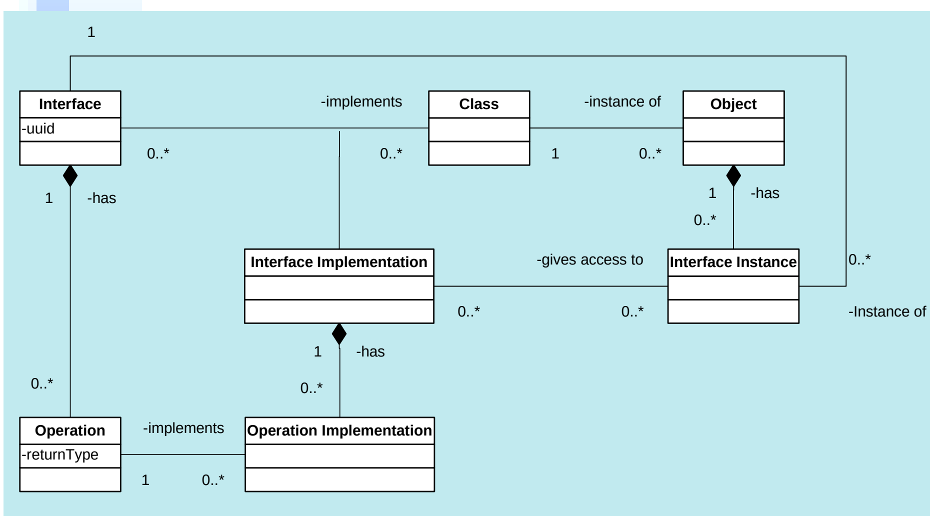
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 44

Interface – Class - Object



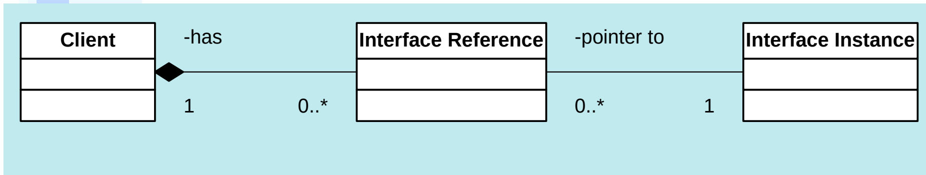
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 45

Interface Instance

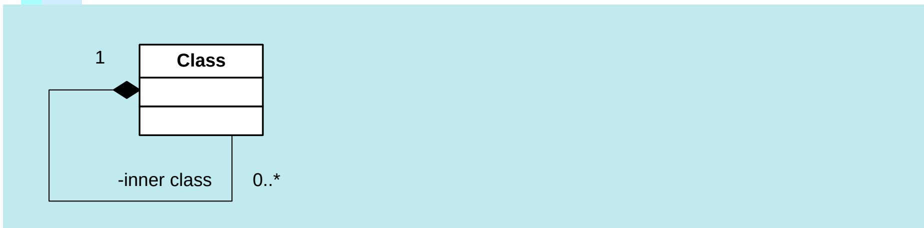


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 46

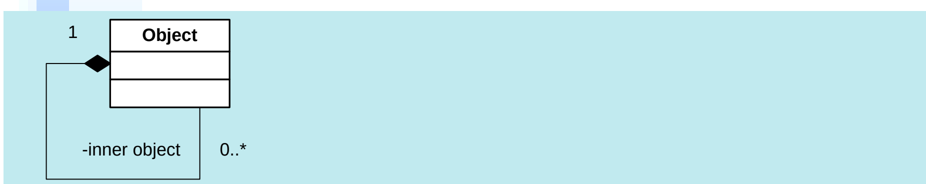
Interface Reference



Inner Class



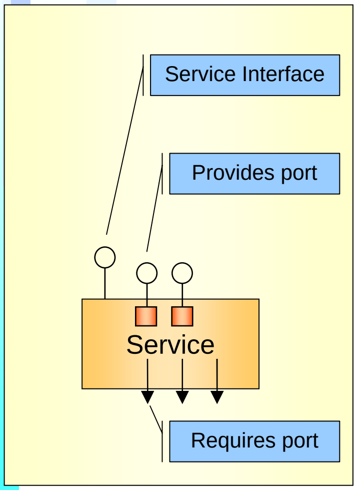
Inner Objects



M3W

Intuition: Service

(Simplified ☺)



Basic Architectural element

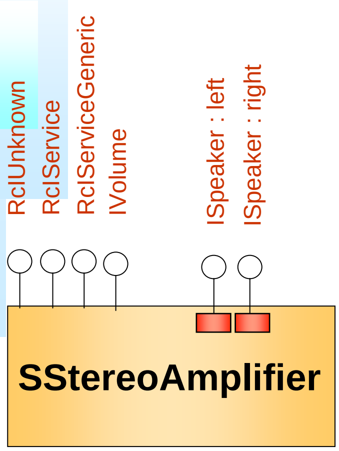
Unit of Instantiation

- Service has 0 or more named Provides ports
- Service has 0 or more named Requires ports
- Ports are of an Interface type
- Interface has 0 or more Operations
- Service implements "Service" Interface
  - Obtain Interface reference to Provides ports
  - Bind Interface references to Requires ports

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 49

M3W

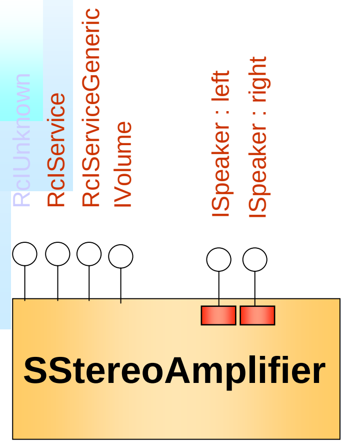
Example: SStereoAmplifier Service




```
service SStereoAmplifier {GUID}
implements IVolume {
   provides{
     ISpeaker left;
     ISpeaker right;
   };
};
```

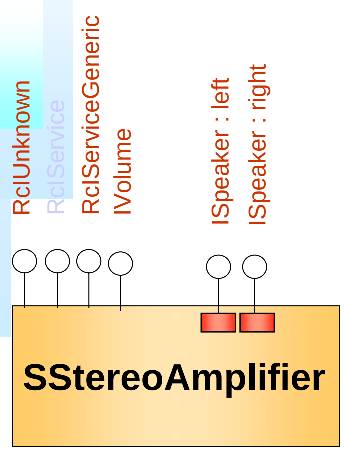
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 50

RcIUnknown



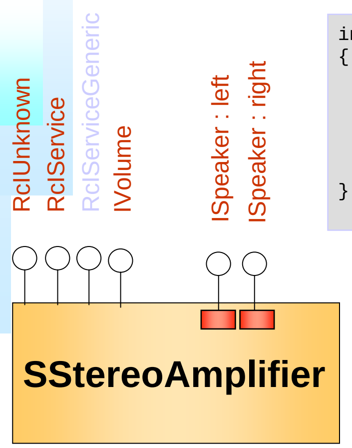

```
interface RcIUnknown { GUID }
{
  native QueryInterface(in guid);
  long AddRef();
  long Release();
};
```


RcIService




```
interface RcIService { GUID }
{
   void start();
   boolean isStarted();
   void stop()
        raises RcXCannotStop;
};
```

Service Specific Interface



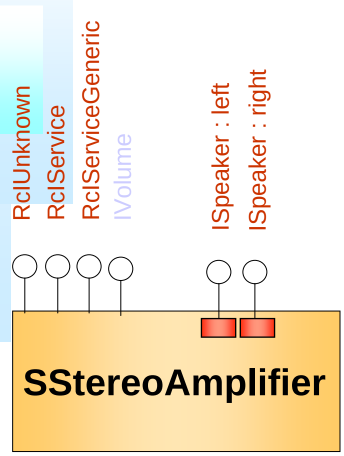

```
interface RcIServiceGeneric { GUID }
{
  IRef getprovides(string nm);
  void setrequires(string nm, IRef pi);
  IRef getrequires(string nm);
  void set(string nm,RcType t,native v);
  native get(string nm,RcType t);
};
```

Used for:

- getting to provided ports
- connecting requires ports
- setting & getting attributes

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 53

Implemented Interfaces



Represents:

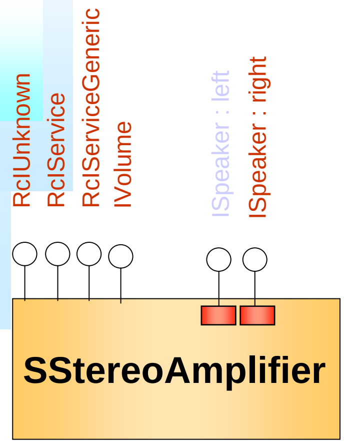
- Is A relations

Properties:

- Navigation between implemented interfaces using QueryInterface()

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 54

Provided Ports



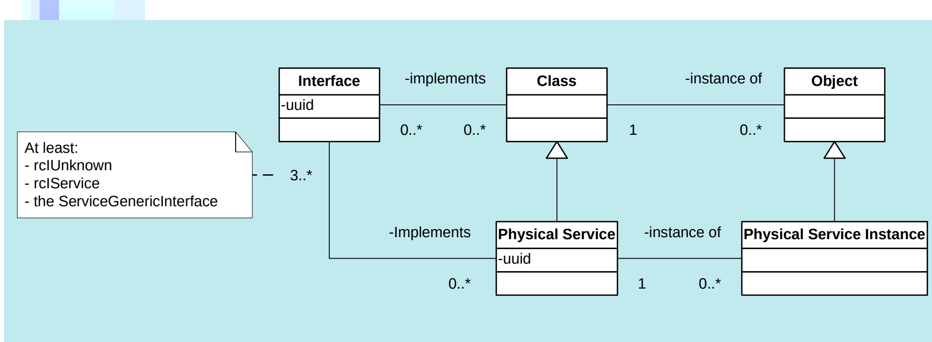
Represents:

- Has A relations

Properties:

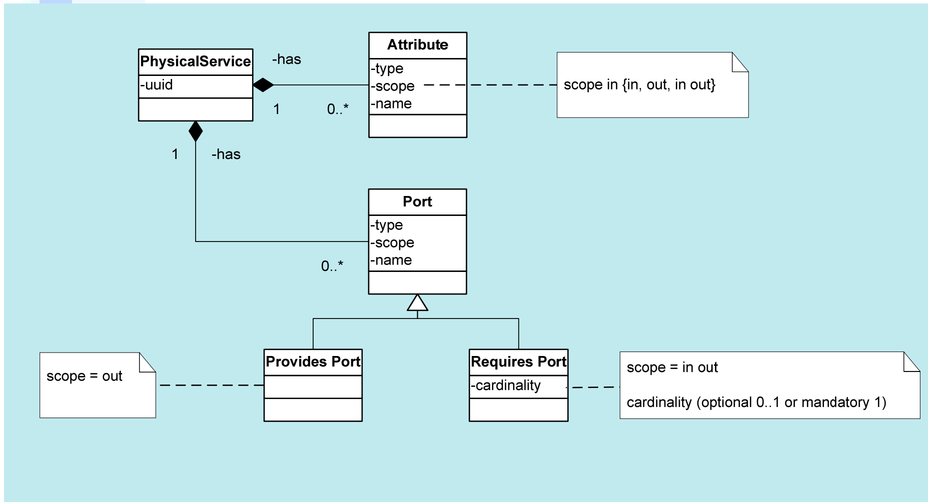
- Navigation between ports is NOT possible
- Reference to ports are obtained using the service specific interface

Class – Physical Service



M3W

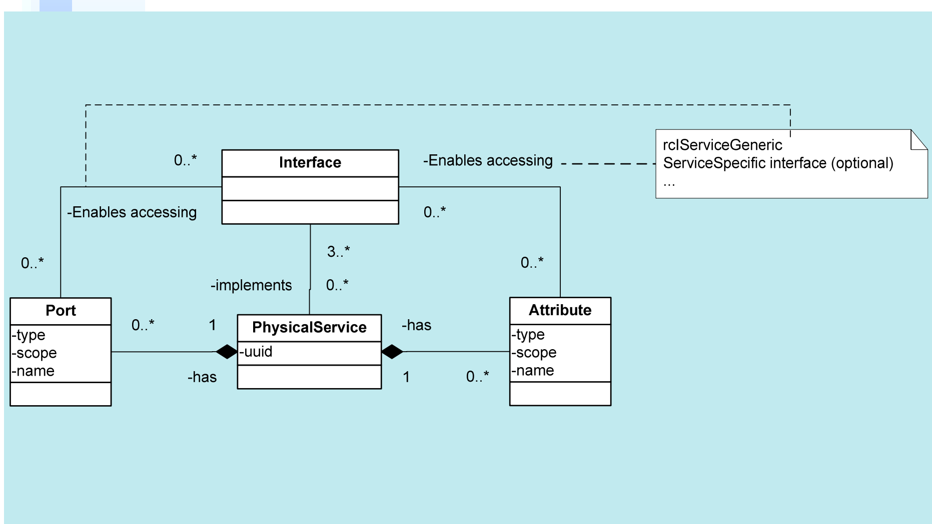
Physical Service and Ports & Attributes



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 57

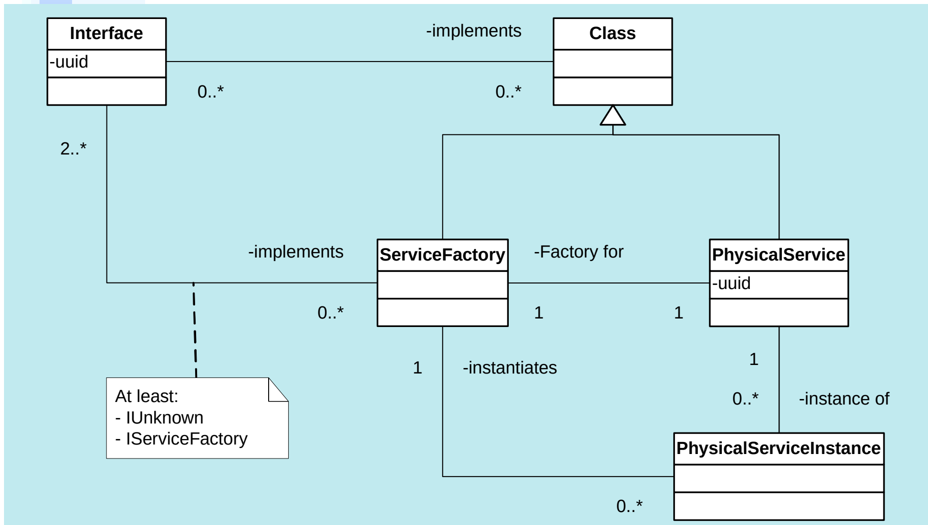
M3W

rcIServiceGeneric Interface



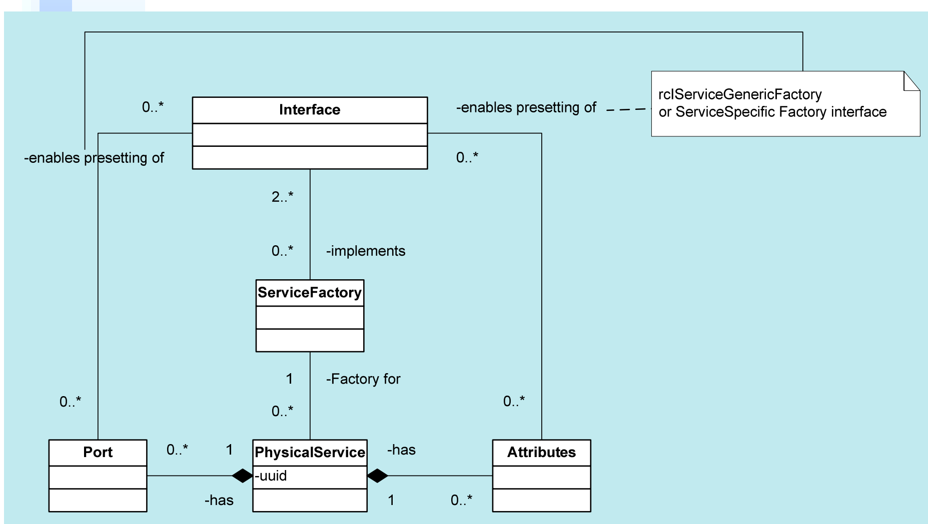
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 58

Service Factory



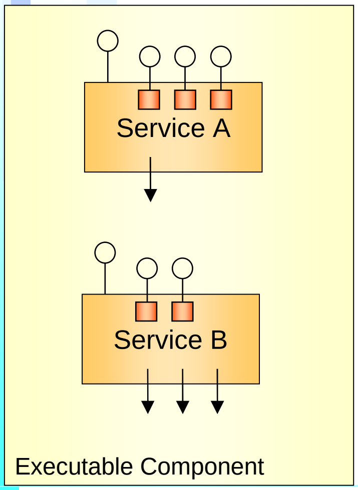
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 59

rcIServiceGenericFactory Interface



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 60

Intuition: Executable Component (Simplified ☺)



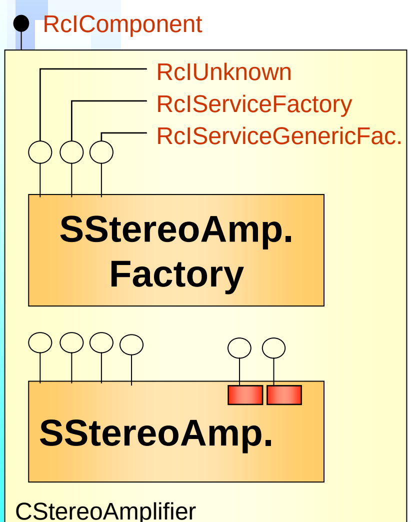
Set of Services

Unit of Loading

- Form depends on the OS, e.g.
  - Static in-process (LIB)
  - Dynamic in-process (DLL)
  - Dynamic out-of-process (EXE)

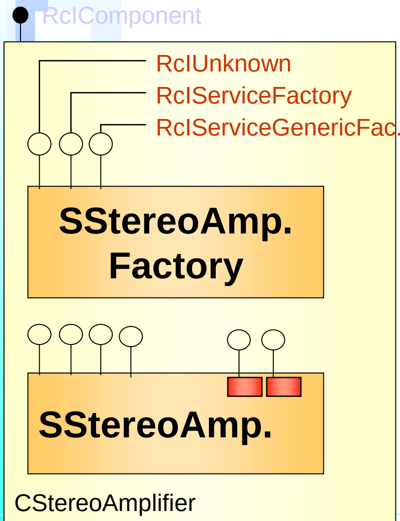
Also contains the “factory” logic for Services Instances, the Service Factory.

Example: Executable Component



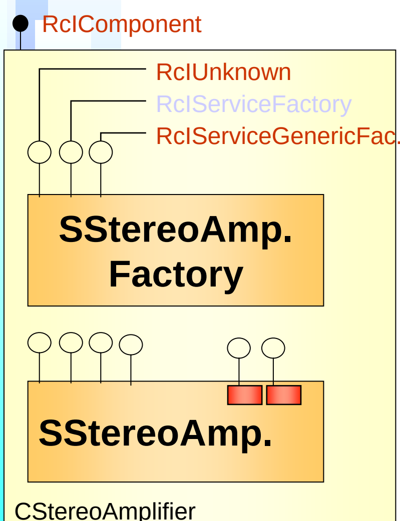

```
component CStereoAmplifier {UUID} {
    provides SStereoAmplifier;
};
```

Example: Executable Component



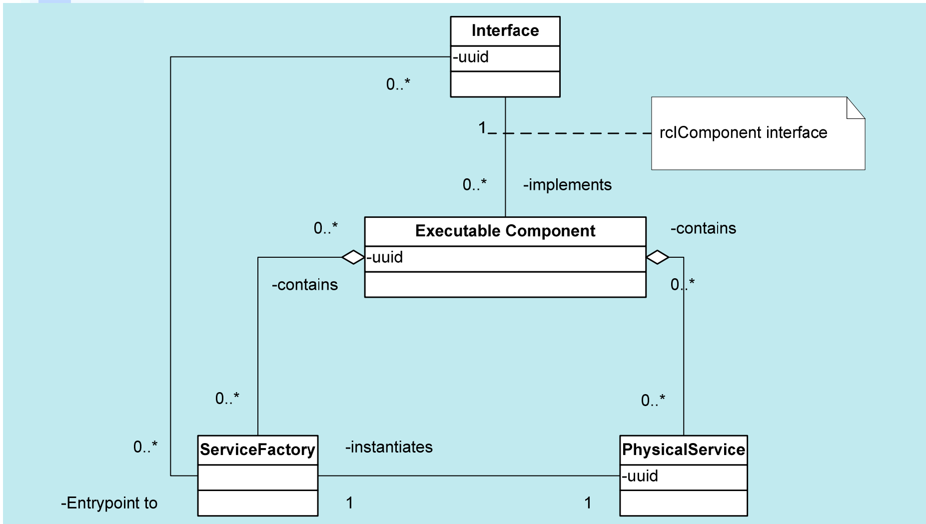

```
interface RcIComponent { UUID }
{
  void initialize();
  boolean isInitialized();
  RcISF getServiceFactory( in uuid
                              svcId);
  void finalize();
  boolean canUnload();
};
```


Example: Executable Component




```
interface RcIServiceFactory { UUID }
{
    RcIService getServiceInstance();
};
```

Executable Component



Component

M3W Component

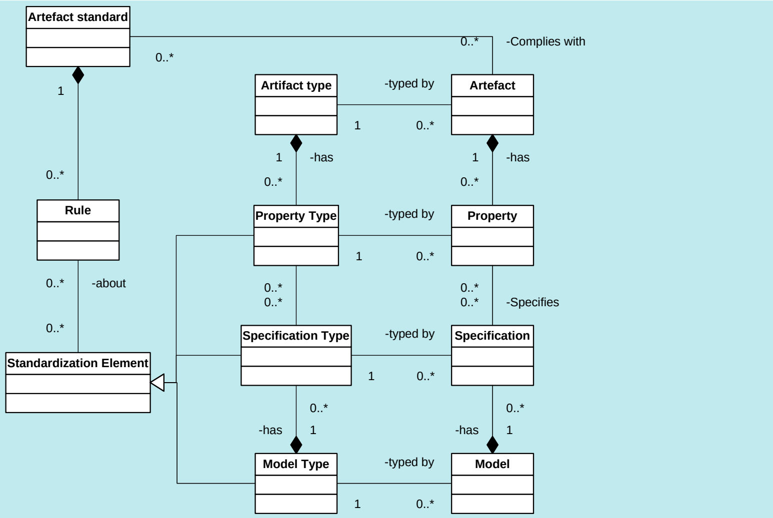
- Resource Model
- Simulation Model
- Documentation
- Functional Model
- Source Code
- ...
- Executable Component

Set of Models & Relations

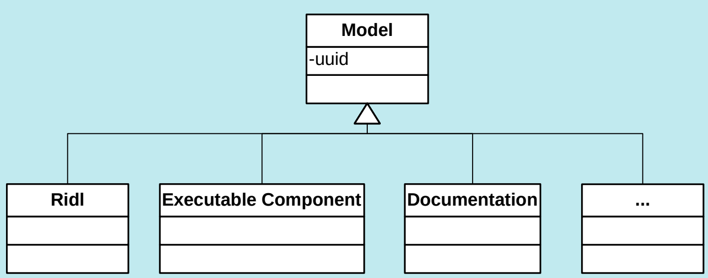
Unit of Trading

- Some model relations defined
- Typically: one (or more) of those Models is executable on a platform

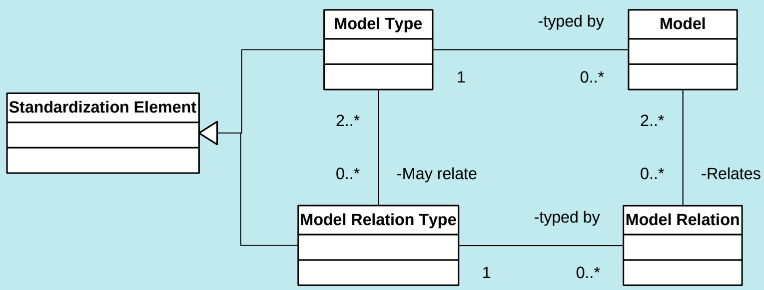
Models



Model Examples

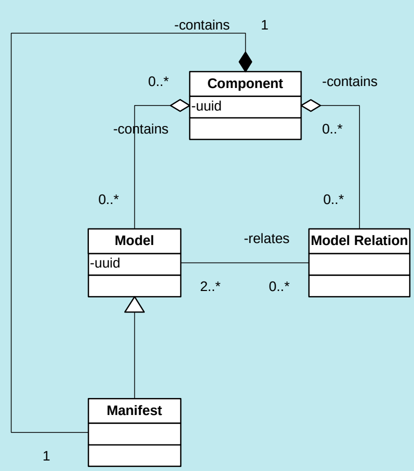


Relations



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 69

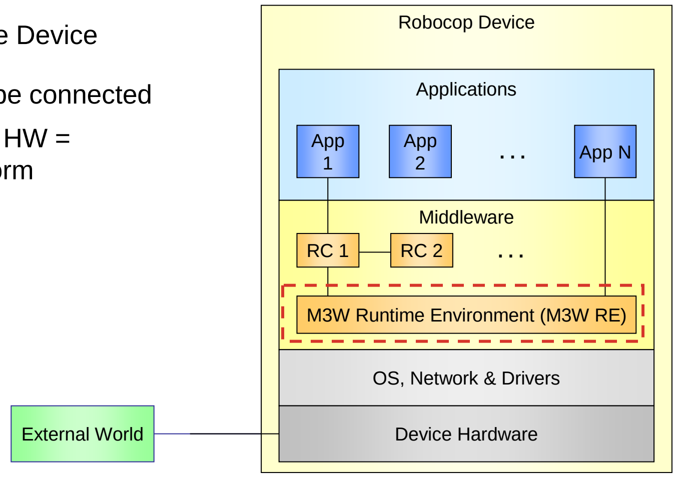
M3W Component



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 70

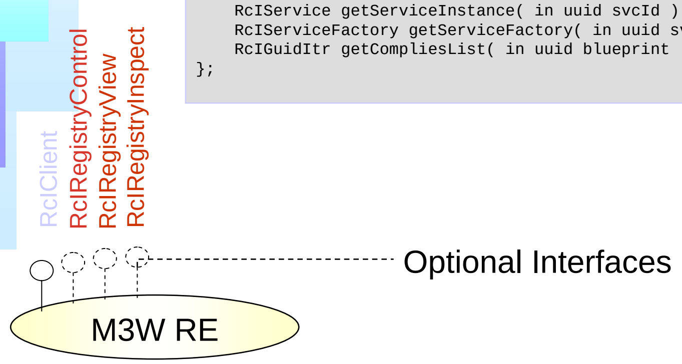
Runtime Environment

- Single Device
- May be connected
- OS + HW = Platform



M3W RE (Instantiation of Physical Services)

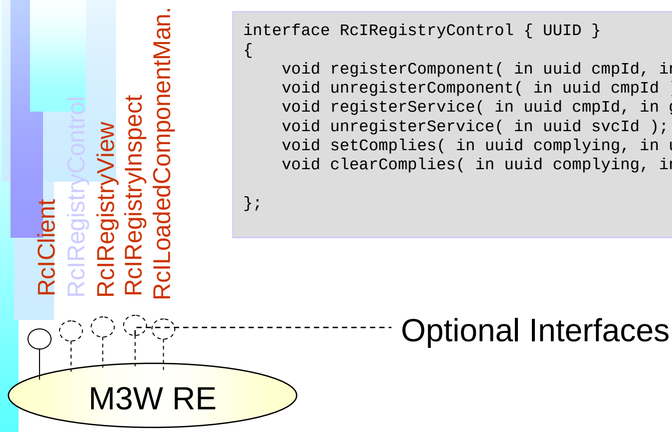
```
interface RcIClient { UUID}
{
    RcIService getServiceInstance( in uuid svcId );
    RcIServiceFactory getServiceFactory( in uuid svcId );
    RcIGuidItr getCompliesList( in uuid blueprint );
};
```



M3W RE (Registration of Components & Services)

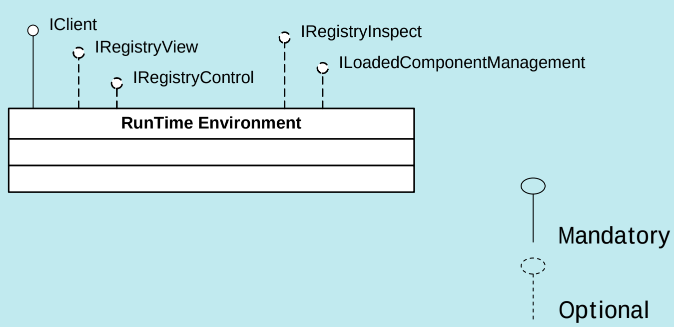
```
interface RcIRegistryControl { UUID }
{
    void registerComponent( in uuid cmpId, in string location );
    void unregisterComponent( in uuid cmpId );
    void registerService( in uuid cmpId, in guid svcId )
    void unregisterService( in uuid svcId );
    void setComplies( in uuid complying, in uuid blueprint )
    void clearComplies( in uuid complying, in uuid blueprint );

};
```



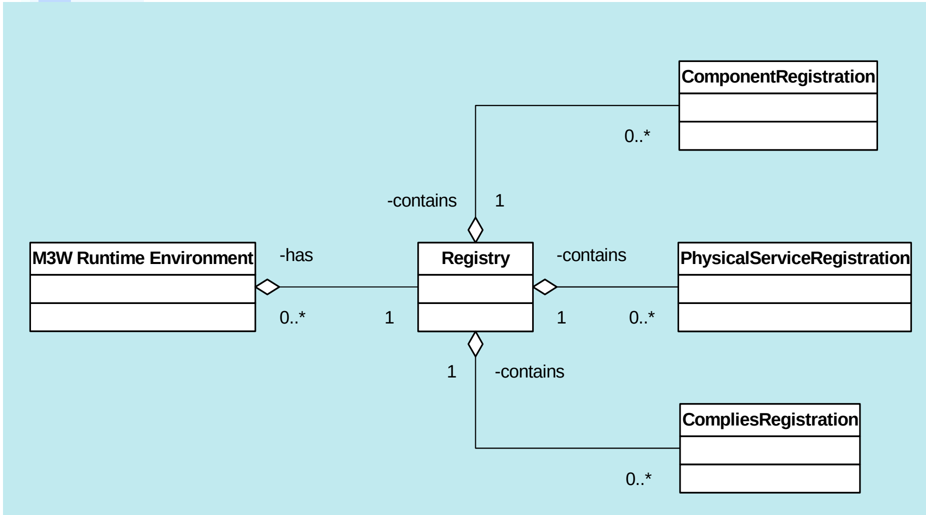
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 73

M3W RE - Interfaces

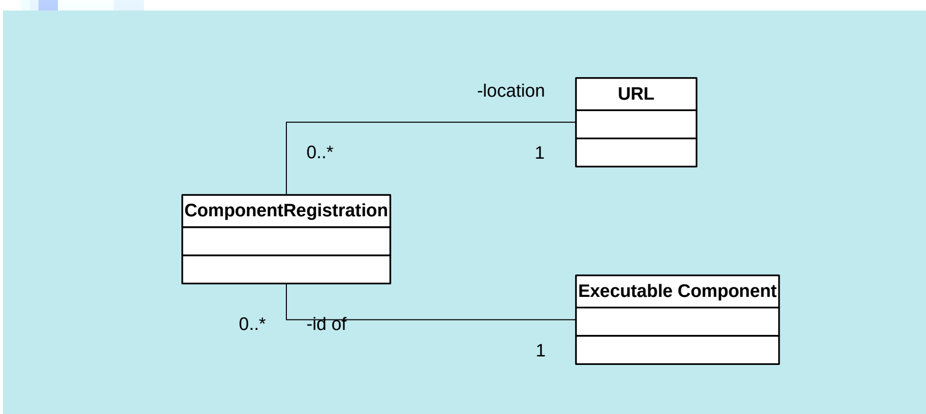


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 74

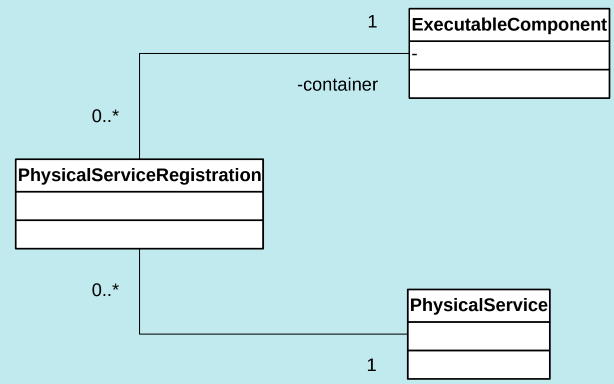
M3W RE - Registry



Component Registration

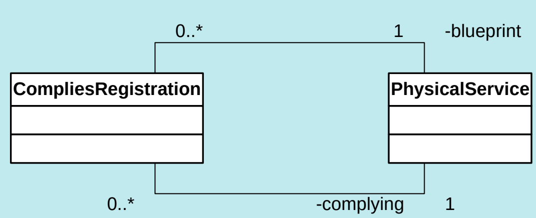


Physical Service Registration



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 77

Complies Registration



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 78

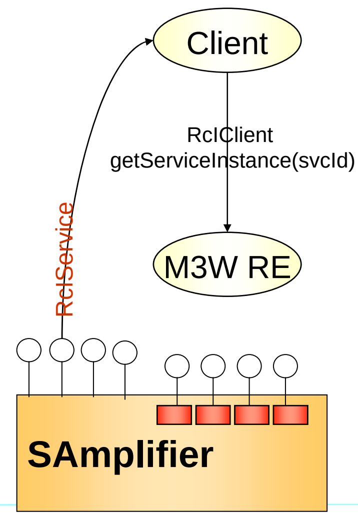
Service Creation using M3W RE

Client

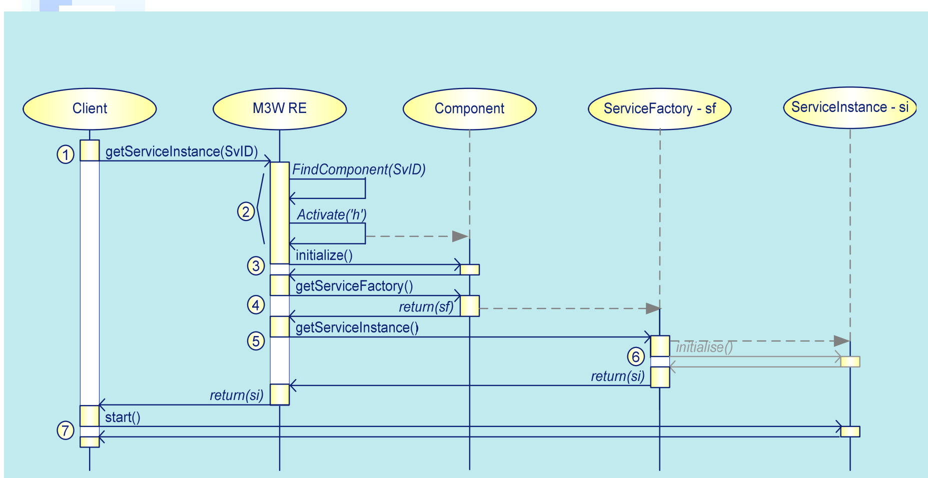
- Asks M3W RE to create a physical service instance
  - Using RclClient Interface
  - Giving UUID of service

RRE

- Create the service instance
- Return a reference to the RclService interface of the created instance



Instantiation using M3W RE



M3W

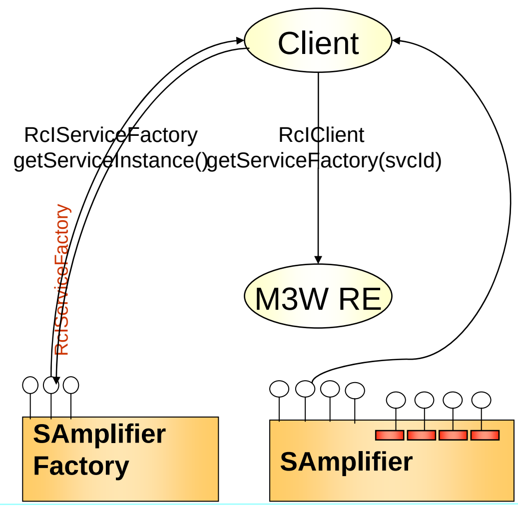
Service Creation using Service Factory

Client

- Asks M3W RE to create a service factory for a phyiscal service
  - Using RclClient Interface
  - Giving UUID of service
- Asks the service factory to create the service instance

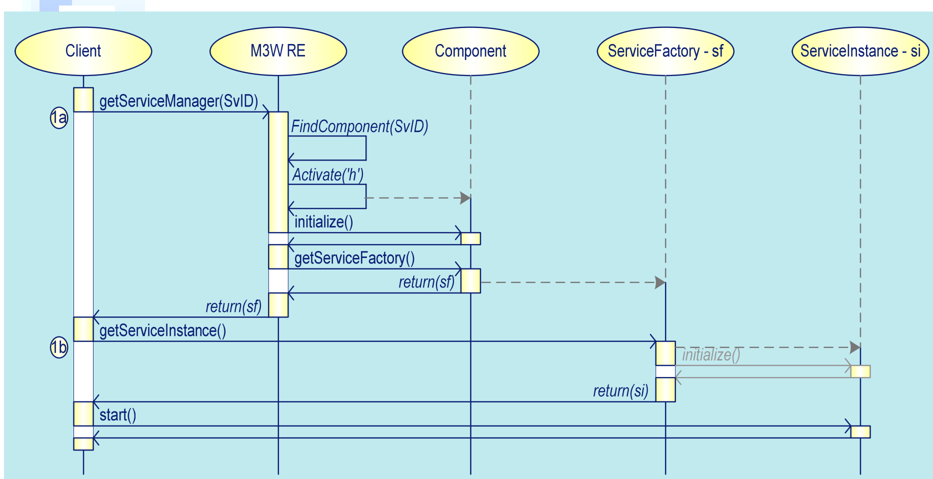
M3W RE

- Create the service factory
- Return a reference to the RclServiceFactory interface of the created instance



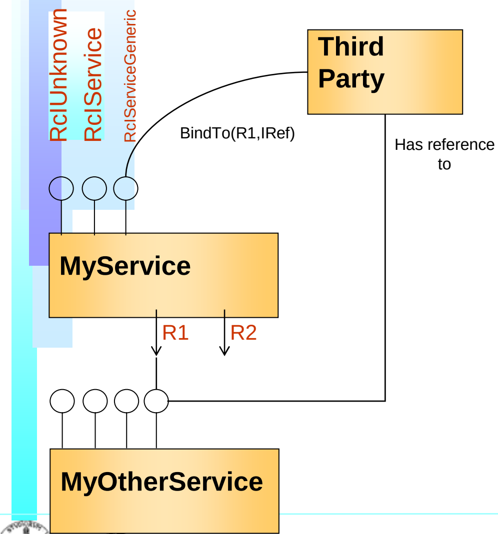
M3W

Instantiation using Service Factory



M3W

Setting Bindings



Third Party

- Binds required ports using the service generic interface implemented by a (physical) service

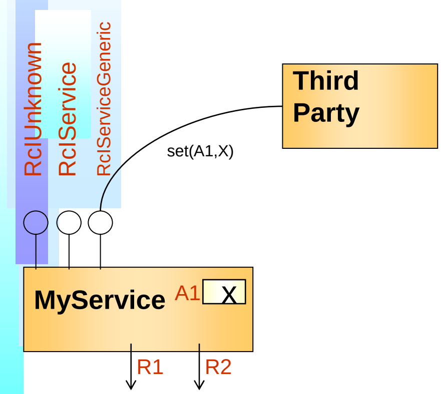
Physical Service

- The `rcIServiceGeneric` interface contains the `BindTo(<name>,<ref>)` operation

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 83

M3W

Setting Attributes



Third Party

- Accesses service attributes using the service generic interface implemented by a (physical) service

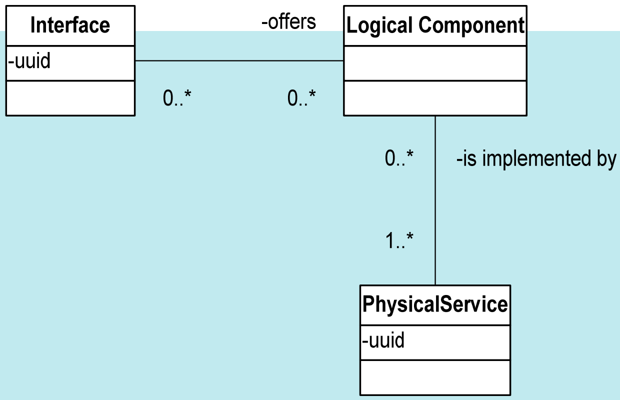
Physical Service

- The service generic interface contains the operation:

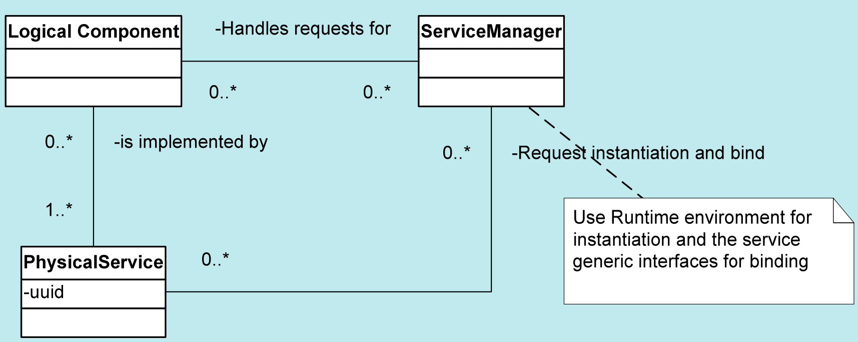
```
set(<attributeName>,<value>)
get(<attributeName>,<value>)
```

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 84

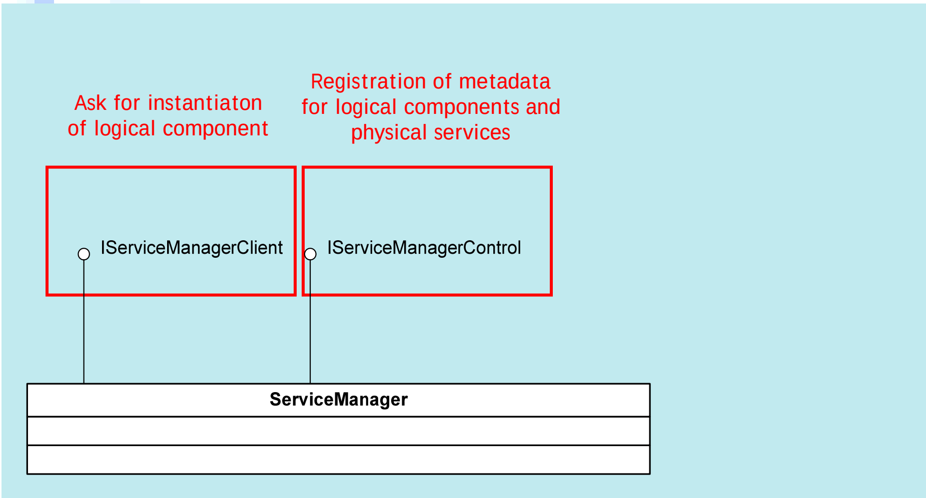
Logical Service



Service Manager



Service Manager Interfaces



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 87

rcISMClient

- Bool isServiceAvailable(uuid logicalComponentId)
- RcIService getInstanceForService(uuid logicalComponentId)

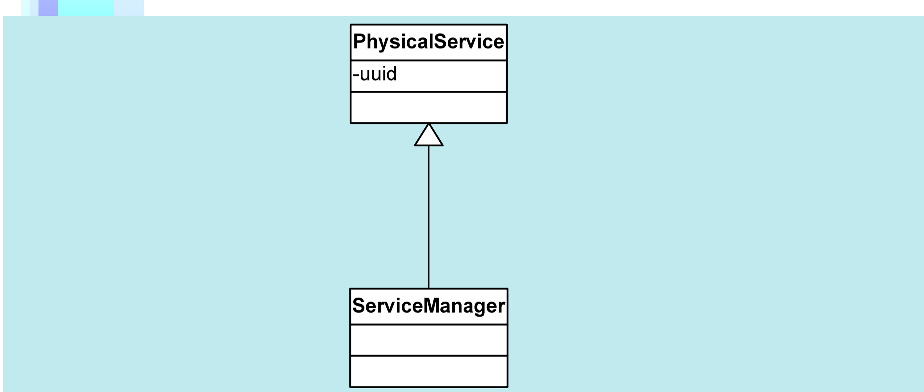
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 88

rcISMControl

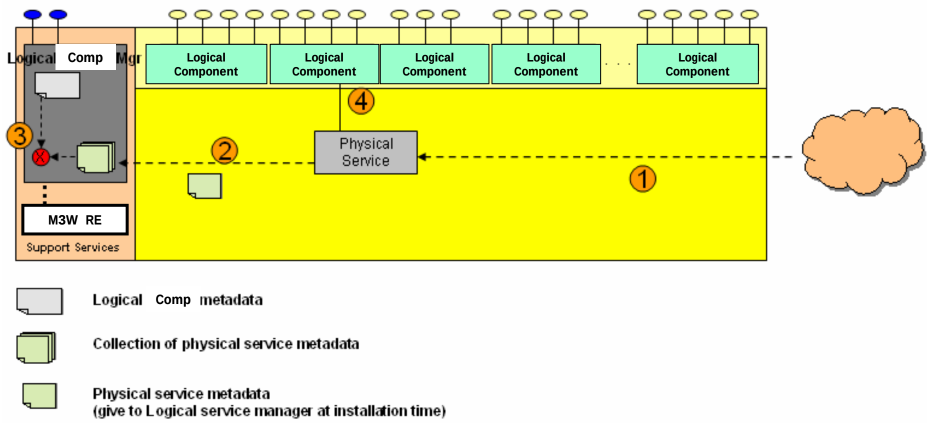
- Bool registerPhysicalServiceMetadata (in String metadataModelPath)
- Bool unregisterPhysicalServiceMetadata (in uuid physicalServiceID)
- Bool registerLogicalComponentMetadata(in String metadataModelPath)
- Bool unregisterLogicalComponentMetadata(in uuid logicalComponentId)

Service Manager

- Service Manager can be implemented as a physical service

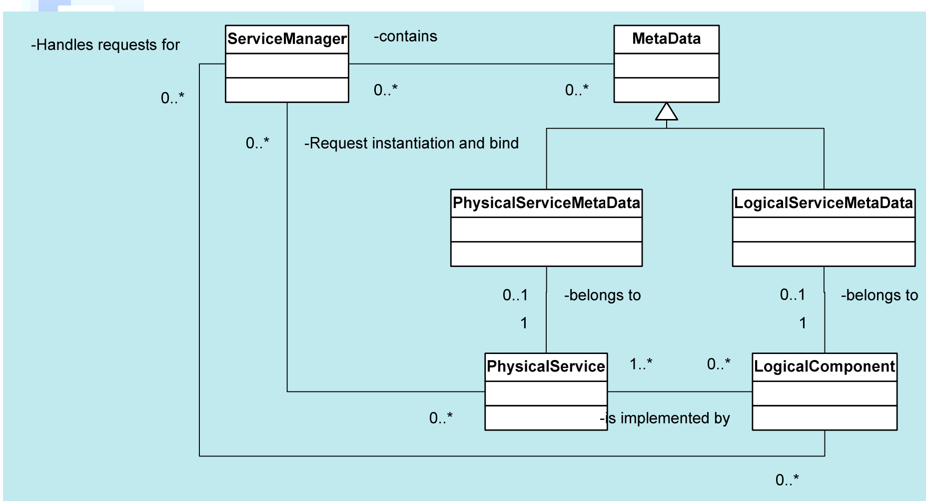


Intuition: Service Manager and Meta Data



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 91

Service Manager and Meta Data



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 92

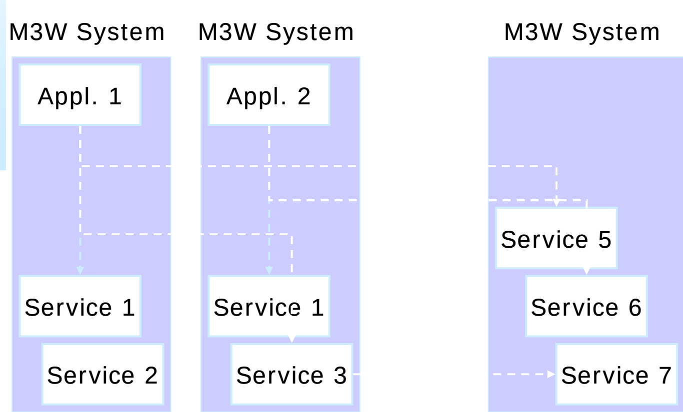
M3W

Remote method invocation

- REMI-R and REMI-P enable
  - Instantiating and using services in remote systems
  - Controlling the life-cycle of remote active instances
- Generated Proxies and Wrappers enable
  - Transparently use of remote services
    - ➔using REMI-R and REMI-P to issue invocation requests

M3W

Use of remote services

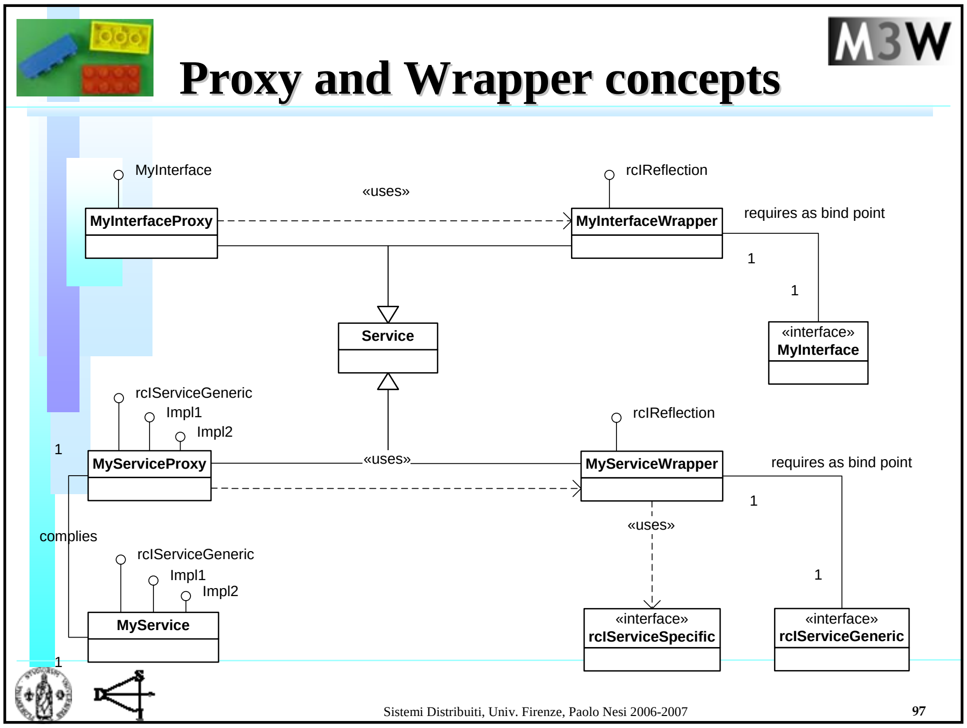


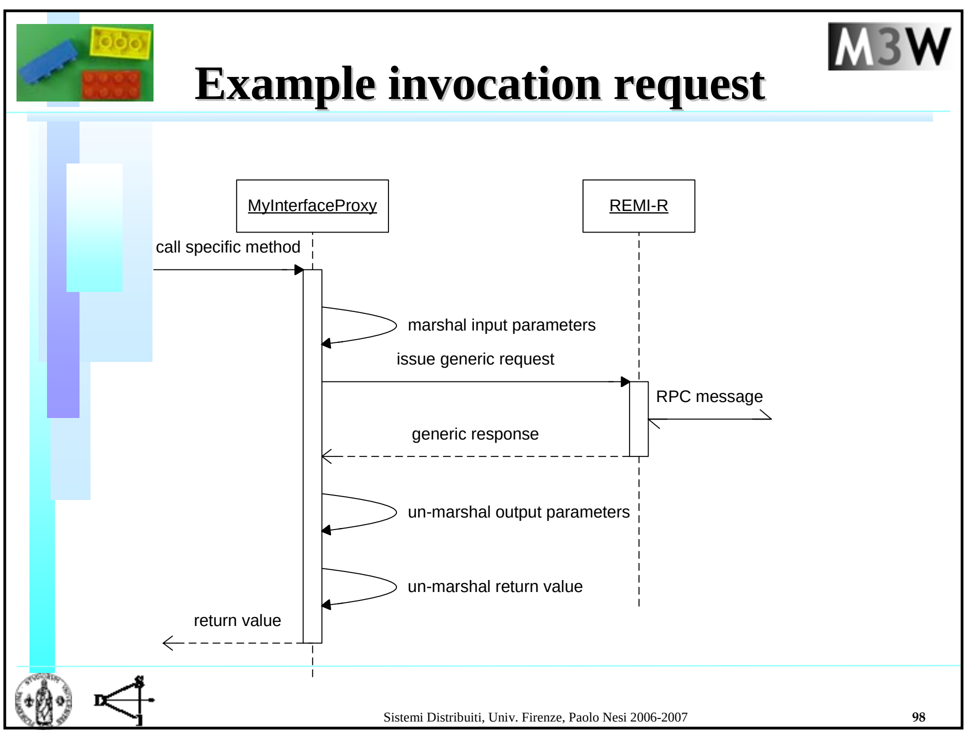
REMI-R responsibilties

- REMI-R manages the forwarding of method execution, creation and releasing of a remote instance
- REMI-R is able to create Proxy instances for a given service
  - Enables transparent usage by the client

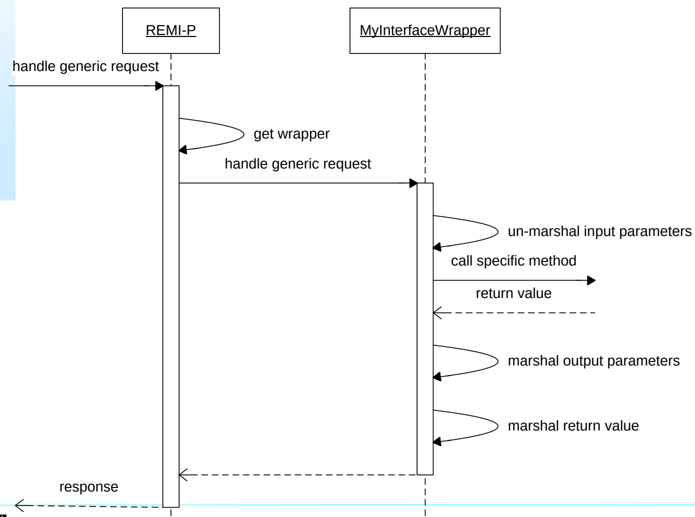
REMI-P responsibilties

- REMI-P manages the incoming requests of method execution, creation and releasing of instances
- REMI-P is able to create Wrapper for a given service
- REMI-P manages the repository of services which are available for remote usage


Proxy and Wrapper concepts
M3W
MyInterface
«uses»
rclReflection
MyInterfaceProxy
MyInterfaceWrapper
requires as bind point
1
1
Service
«interface»
MyInterface
rclServiceGeneric
Impl1
Impl2
rclReflection
1
MyServiceProxy
«uses»
MyServiceWrapper
requires as bind point
1
«uses»
complies
rclServiceGeneric
Impl1
Impl2
1
MyService
«interface»
rclServiceSpecific
«interface»
rclServiceGeneric
1
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
97


Example invocation request
M3W
MyInterfaceProxy
REMI-R
call specific method
marshal input parameters
issue generic request
RPC message
generic response
un-marshal output parameters
un-marshal return value
return value
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
98

Example handling of invocation request



Resource Management Framework

<u>Objectives:</u>

- Robustness in the time domain
- Resource management of
  - memory, processor, bus, network, …
- Guaranteed execution of tasks with time requirements
- Quality of Service Management

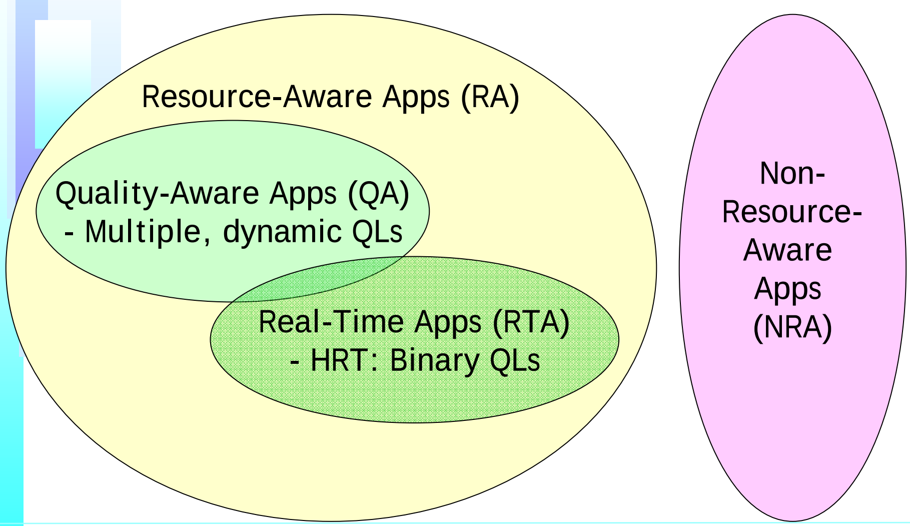
Some RM definitions

- Resource-aware (RA) applications:
  - Know their resource needs.
- Quality-aware apps (QA): Are RA apps
  - Provide a number of output quality levels (QL)
  - Can change their quality level dynamically
- Real-Time apps (RTA): Are RA apps
  - Applications with time requirements.
  - To guarantee them, know required resources
  - HRT apps. provides two QLs: all and nothing

NOTE

The word “app” in this context is a collection of Service Instances and/or Applications that form a logical whole from RM point of view

Resource awareness

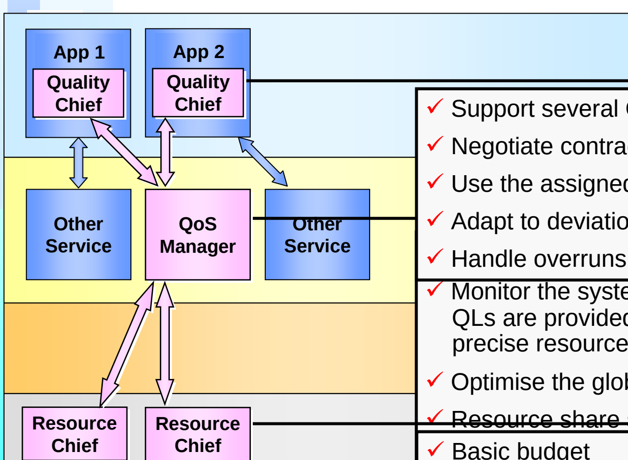


Approach to Resource & Quality Management

- Based on a contract model
  - Resource Management Framework provides resource
  - Applications provide a certain Quality Level
- Negotiation based
  - Applications provide <quality level, resource needs> options
  - The RMF (selects the option and) assigns resources to Resource Aware applications.
  - ***A portion of the available resources is reserved for Non-Resource Aware apps***

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 103

Resource Management Architecture

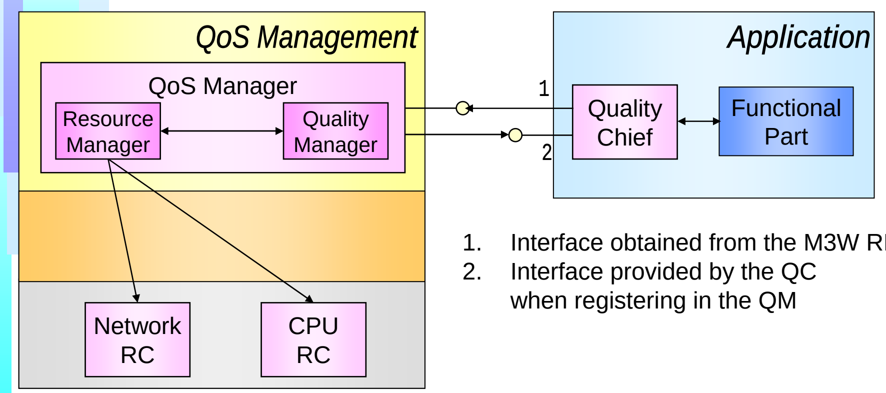


- ✓ Support several QLs
- ✓ Negotiate contract with QoSM
- ✓ Use the assigned budgets
- ✓ Adapt to deviations.
- ✓ Handle overruns and errors
- ✓ Monitor the system to ensure that QLs are provided & to know precise resource usage
- ✓ Optimise the global quality
- ✓ Resource share setting
- ✓ Basic budget
- ✓ Events for accounting & enforcing

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 104

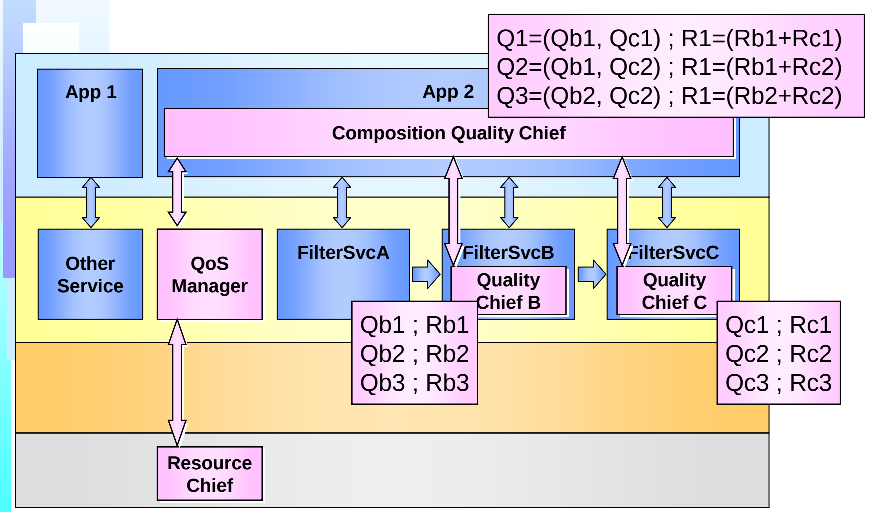
Interaction Model

QoS manager is a M3W Physical Service



1. Interface obtained from the M3W RE
2. Interface provided by the QC when registering in the QM

Composition of QoS Information

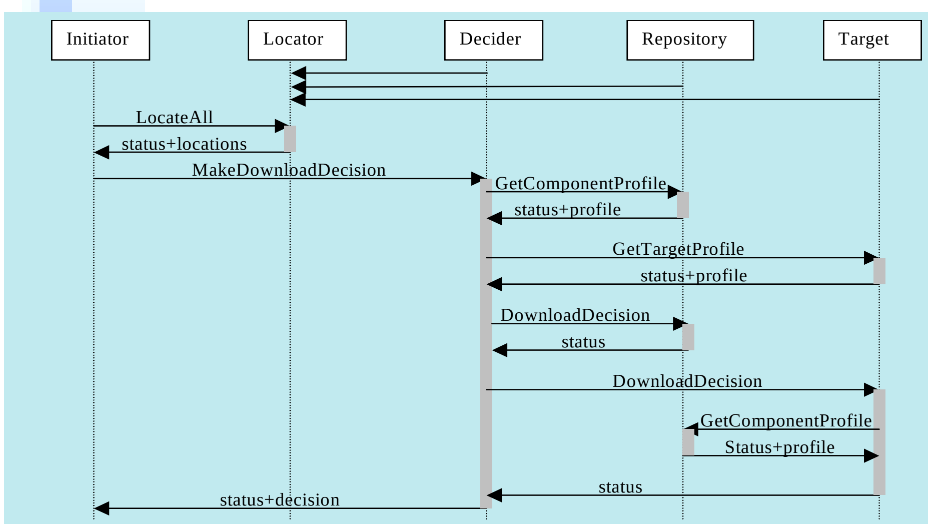


Download Framework

- Objective: controlled download of Components
- Entities
  - Repository: where component to be downloaded resides
  - Target: device where the component will be downloaded to
- Roles
  - Initiator: identifies the need for a download and contacts the involved parties to initiate the process
  - Locator: locates Target, Repository and Decider for a download
  - Decider: performs the feasibility analysis for the download:
    business fit & technical fit & resource fit

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 107

Process Steps (1, negotiation)



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 108

Decision is based on Profiles

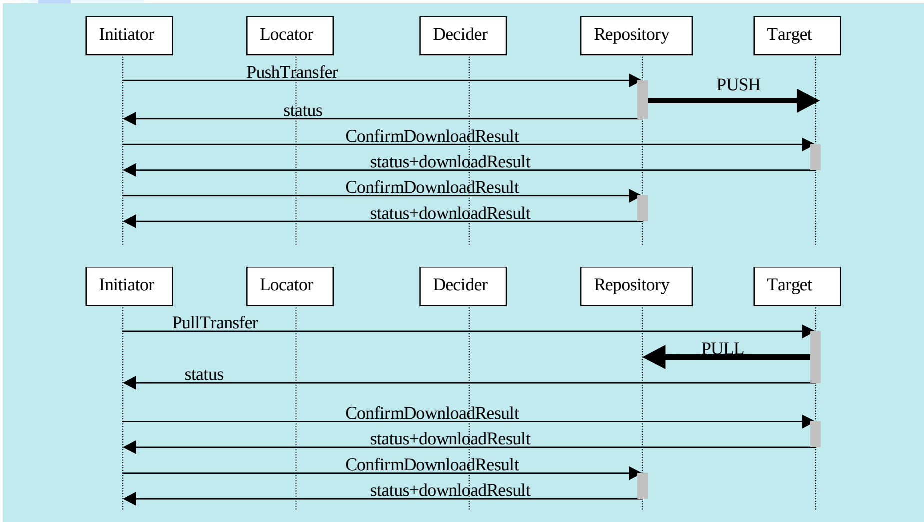
Component profile *compatible with* Target profile

- *Target profile*: snapshot of the Target's state and properties
- *Component profile*: the component's requirements for
  - installation on the Target
  - execution on the Target
- If -given the Decider's rules- both profiles are matching, the Download can proceed. If not, another component can be selected in the list gathered from the repository

Profile attributes

Component profile	Target profile
Number of bytes transferred to Target for component installation	Available memory on the device
Stored component size	Id.
Installed component size	Id.
Initialized component size	Id.
Instantiated component size	Id.
Maximum amount of dynamic memory required by a component instance	Id.
CPU information for which the code was generated	Device CPU-information
Number of bytes of persistent memory needed by the component	Available memory on the device
Specific hardware requirements (e.g., clock, display, MPEG decoder),	Specific hardware features
Specific services of the OS (like semaphores, process priorities, timers, etc.)	OS information (maker, version, type, capabilities, services, etc.)
Target platform/operating system (family, etc.) for which the component is suited (can be manufacturer dependent),	Id.
Platform/operating system versions acceptable for the component,	Id.
List of specific platform services (not specifically linked to M3W), e.g. a communication stack required by the component	List of specific platform services
Transferred file format (e.g., zip, jar, ...),	List of supported transfer formats
Executable file format (e.g. ELF),	List of supported executable formats
Language of the installed component code (and maybe the version of the language)	List of supported development languages
Versioning information like some major/minor component version number,	Supported versions, families, etc.
CPU/thread usage, expressed in terms of required QoS.	Available resources
Trading information, like a manufacturer identification,	Identification, signature

Process steps (2 - PUSH & PULL models)



Download API

- The Download Process is accessed through an API
- One API for each role / entity:
  - Locator API
  - Decider API
  - Initiator API
  - Repository API
  - Target API
- General characteristics of the API :
  - Blocking calls.
  - RPC-like syntax so that it can be projected to RMI, Web services, etc.
  - Transparently local or distant.

Fault Management Framework objectives

M3W

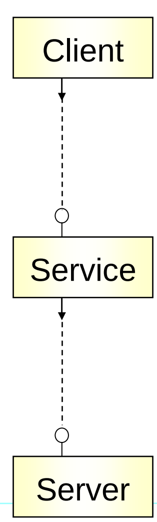
- Ability to make systems composed of M3W Service Instances fault tolerant.
  - Fault-manage (instances of) “black box” Service Instances
- Be transparent to creator, server, and the Service Instance being fault-managed
- Ability to co-ordinate Fault handling between multiple Service Instances

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 113

Fault Management - Initial Situation

M3W

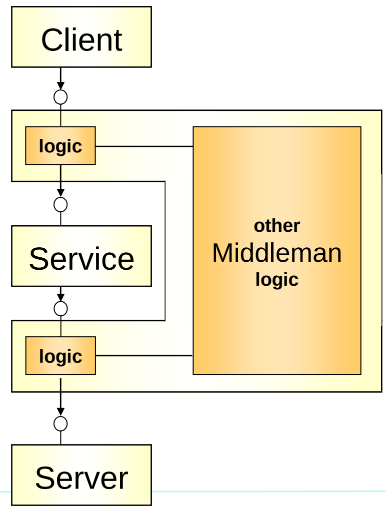
- Client uses the Service
- Service uses a Server
- Binding through Interfaces
- Service may be “untrusted”



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 114

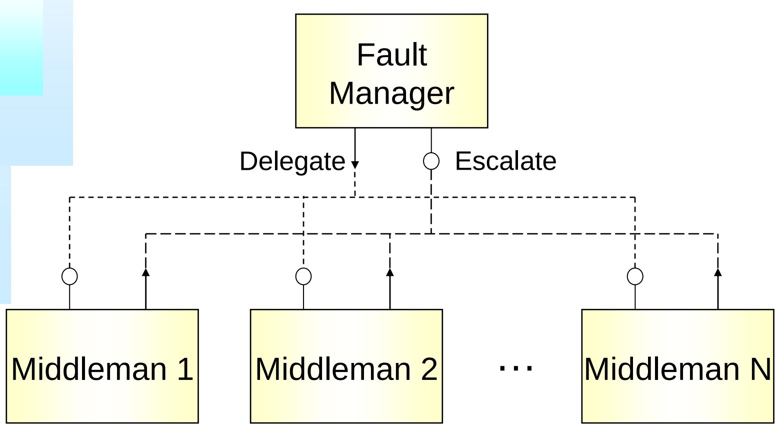
Fault Management - Managed Situation

- Intercept creation of the Untrusted Service Instance
  - Fault Management policy
- Replace with Middleman
  - Middleman may use the Untrusted Service Instance
  - May intercept Interface Bindings
  - May contain other logic



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 115

Fault Management – Coordinating Multiple MM



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007 116

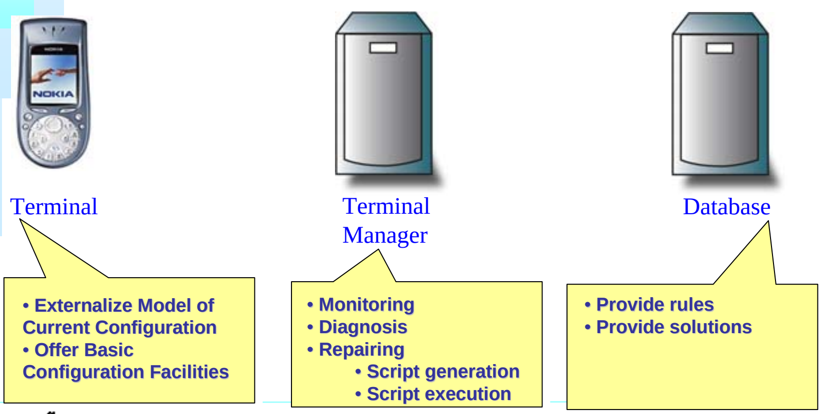
System Integrity Management Objectives

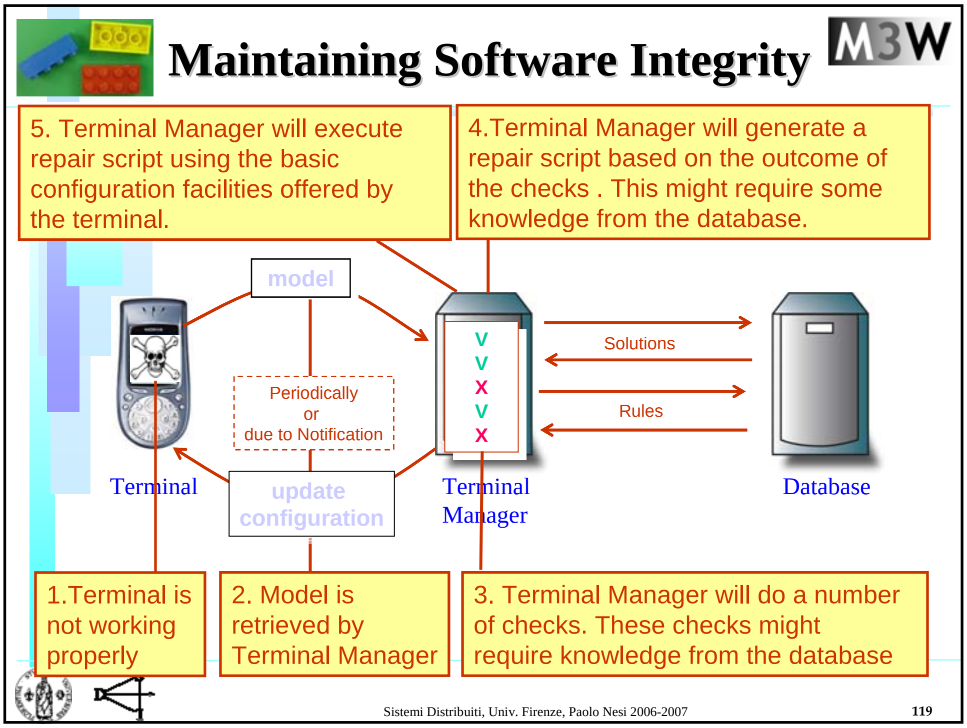
- Maintain a terminal in a consistent & sane state on behalf of the user and/or service provider
  - Also in the view of new information becoming available
- Manage upgrades & updates
  - Also based on context information (e.g. bus-stop)
- Provide a “reporting point” for fault management

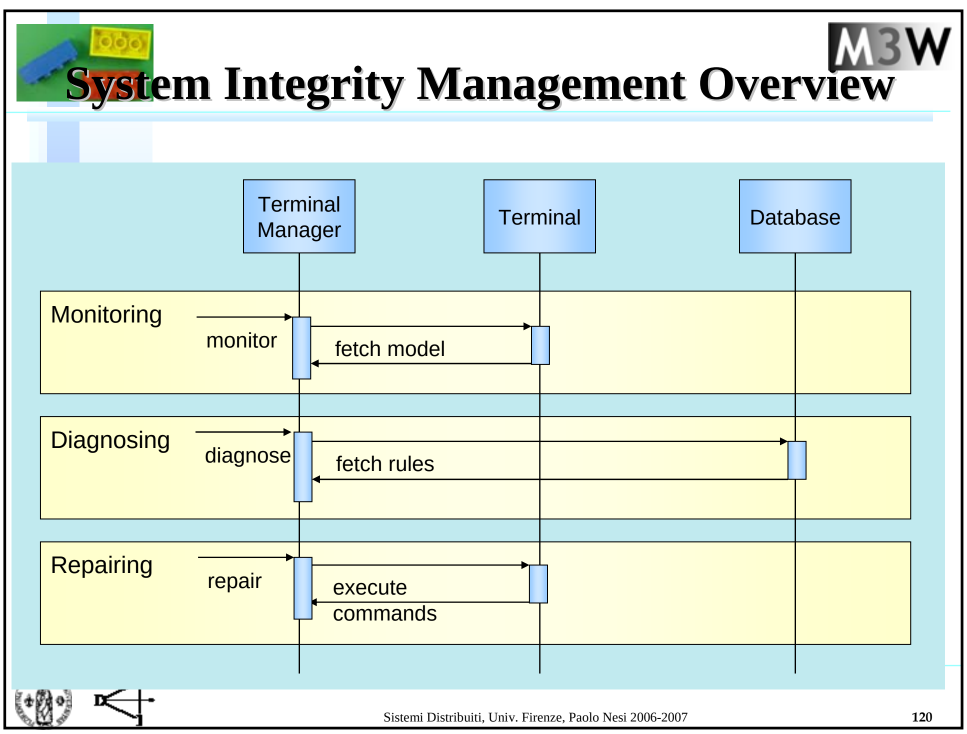
Maintaining Software Integrity

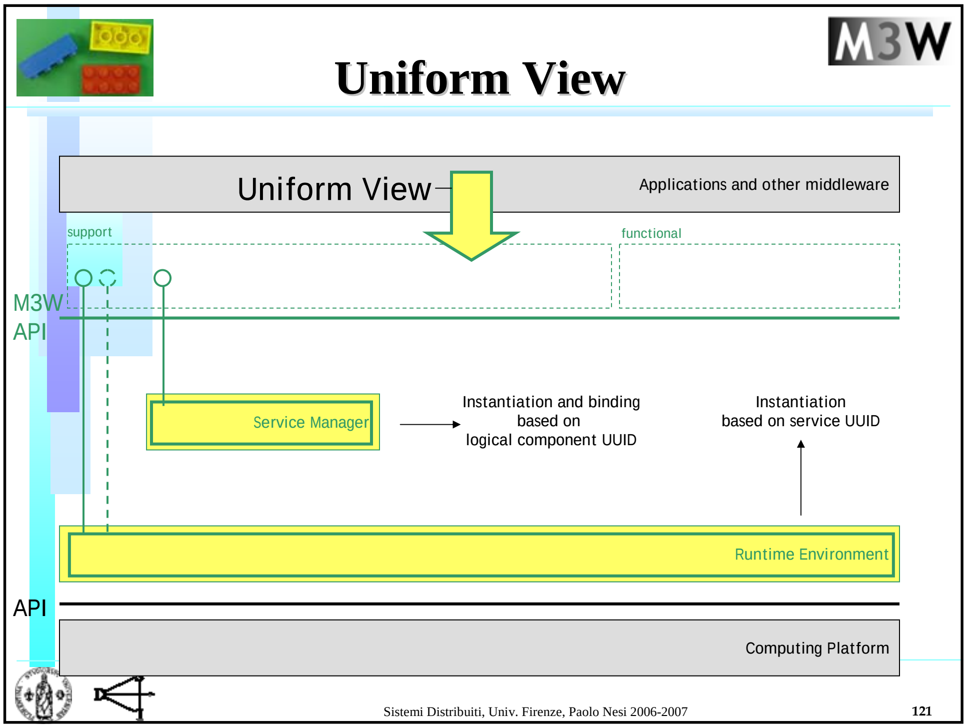
Approach: Maintaining software integrity using 3 roles !

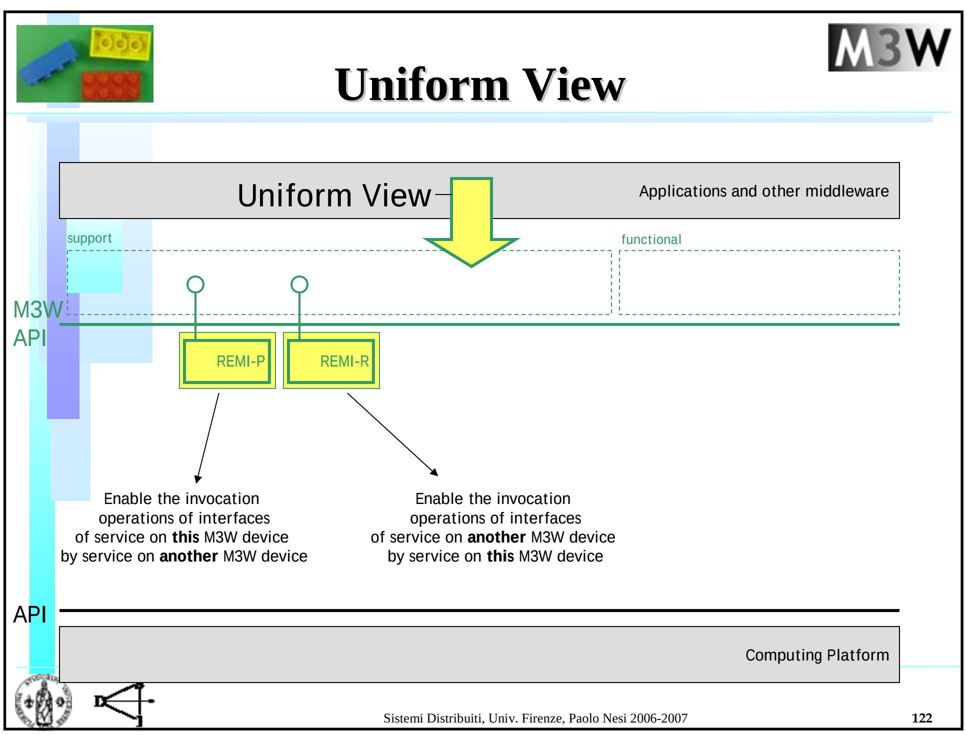
Responsibilities: of the individual roles …



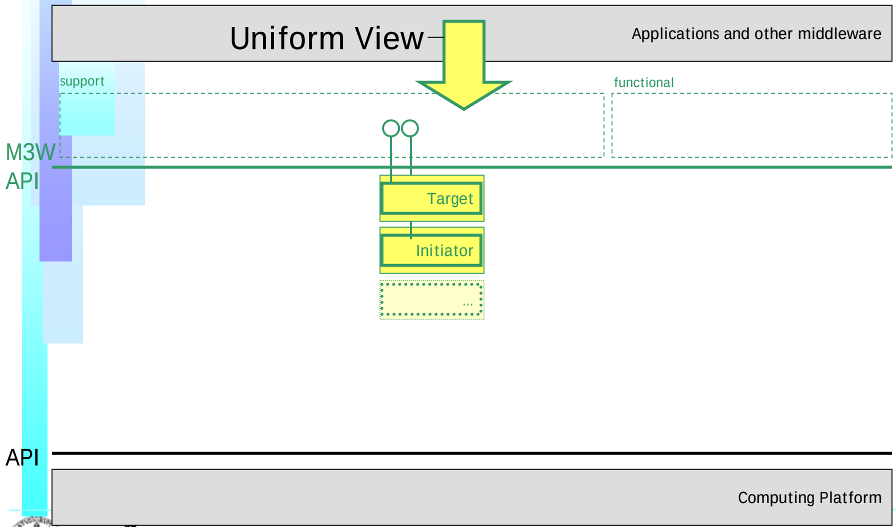

Maintaining Software Integrity
M3W
5. Terminal Manager will execute repair script using the basic configuration facilities offered by the terminal.
4.Terminal Manager will generate a repair script based on the outcome of the checks . This might require some knowledge from the database.
model
Periodically or due to Notification
update configuration
V
V
X
V
X
Solutions
Rules
Terminal
Terminal Manager
Database
1.Terminal is not working properly
2. Model is retrieved by Terminal Manager
3. Terminal Manager will do a number of checks. These checks might require knowledge from the database
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
119


System Integrity Management Overview
M3W
Terminal Manager
Terminal
Database
Monitoring
monitor
fetch model
Diagnosing
diagnose
fetch rules
Repairing
repair
execute commands
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
120


M3W
Uniform View
Uniform View
Applications and other middleware
support
functional
M3W API
Service Manager
Instantiation and binding based on logical component UUID
Instantiation based on service UUID
Runtime Environment
API
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
121


M3W
Uniform View
Uniform View
Applications and other middleware
support
functional
M3W API
REMI-P
REMI-R
Enable the invocation operations of interfaces of service on this M3W device by service on another M3W device
Enable the invocation operations of interfaces of service on another M3W device by service on this M3W device
API
Computing Platform
Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2006-2007
122

Uniform View



References

- M3W http://www.chiariglione.org/mpeg/technologies/mpe-m3w/index1.htm