System integration based on middleware

What is it and how is it used?

Alberto Messina

RAI – Centro Ricerche e Innovazione Tecnologica

EBU NMC Seminar,

Geneva 16th and 17th June 2004

Purpose of the presentation

- ✓ The presentation will show some results concerning our work in P/MDP (Middleware in Distributed Production)
- ✓ Try to give a theoretical view that helps us to understand the problem of system integration
- ✓ Purposes
  - ☞ Express our ideas better, with the right terminology, avoiding common misunderstandings and hypes
  - ☞ Identifying the right tools and methods for the job

Roadmap

1) A theoretical view on Systems' structures

    ☞Layered systems

    ☞Open and closed systems

    ☞Services

✓ 2) A theoretical view on Systems' integration

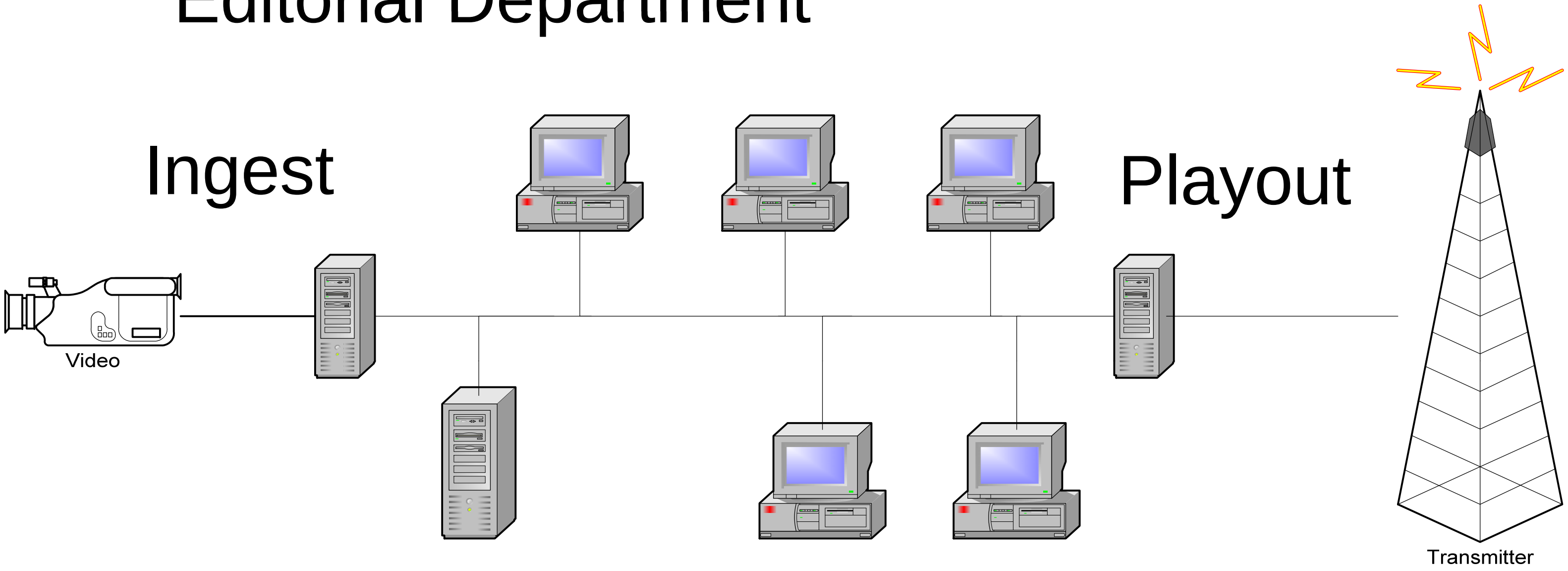
    ☞Types of integration

    ☞Where middleware does come up

✓ 3) Enabling tools & technologies

✓ 4) Conclusions

A future scenario?



Editorial Department

Ingest

Playout

CMS

Editing

Common misunderstandings

- ✓ Some hypes:
  - ☞ Middleware is a technology that solves all system integration problems
  - ☞ Middleware is an off-the-shelf solution
  - ☞ Damn! You don’t have it yet! Do you wanna buy?
- ✓ Some facts:
  - ☞ The term “middleware” is a loose term, the right topic to be investigated is **system integration**
  - ☞ System integration is something we do each day, so nothing is new under the sun appearently
  - ☞ There are the enabling methods and technologies that help us in this activity

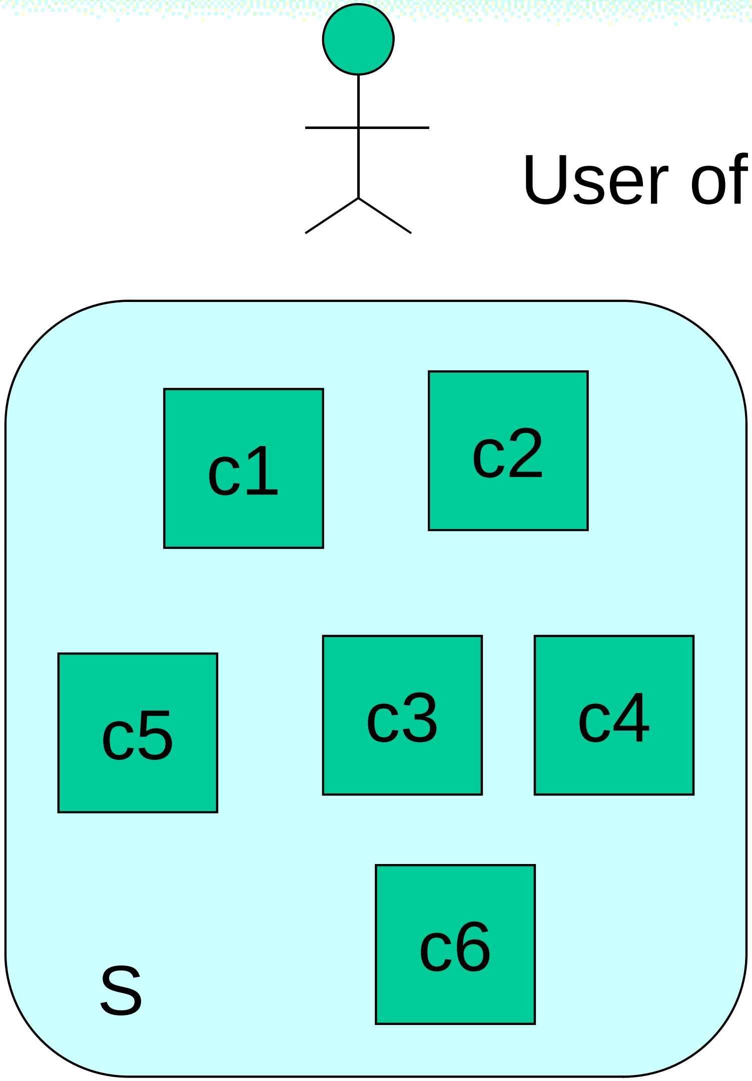
Some fundamentals

- ✓ A system is something purposed to satisfy a determined use
  - ☞ Example: VTRs are objects **used** to record and play audiovisual material
- ✓ Any system can be viewed as an organised group of *components*
  - ☞ Example: a VTR is made up of a command console, a set of rec/play heads, a set of output and input plugs
- ✓ What makes a system look like as such can depend on whom is looking at the system
  - ☞ Example: to a maintainer a VTR seems to have much more components than to an editor!

What are components features?

- ✓ A component of a system is characterised by its interface, i.e.:
  - ☞ An unique identification
  - ☞ A set of operations which can be invoked on the component
  - ☞ A set of (I/O) parameters for each of the operations
  - ☞ A well-defined behaviour
  - ☞ A well defined semantics

A simple picture explanation



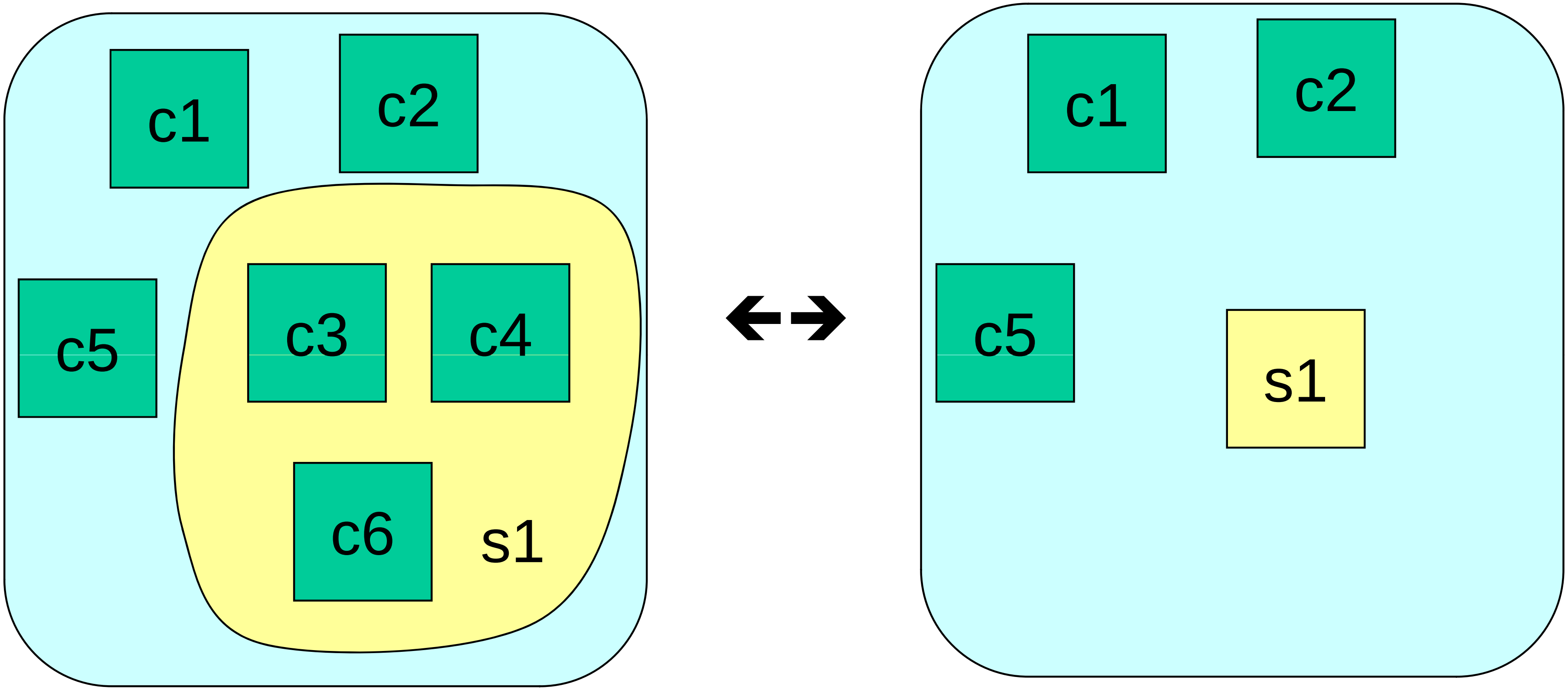
User of class U

Components seen by any user of class U

Subsystems

- ✓ We can define subsets of components for the system
- ✓ We can call these subsets *subsystems*
- ✓ A subsystem is characterised by its interface as well:
  - ☞ An unique identification (other than any of its constituent components')
  - ☞ A set of access operations that is the union of the operations of its components
  - ☞ A set of (I/O) parameters for each of the operations
  - ☞ A well-defined behaviour
  - ☞ A well defined semantics
- ✓ It follows that subsystems *are* components
  - ☞ We will always use "components" unless specified to indicate both atomic components and subsystems

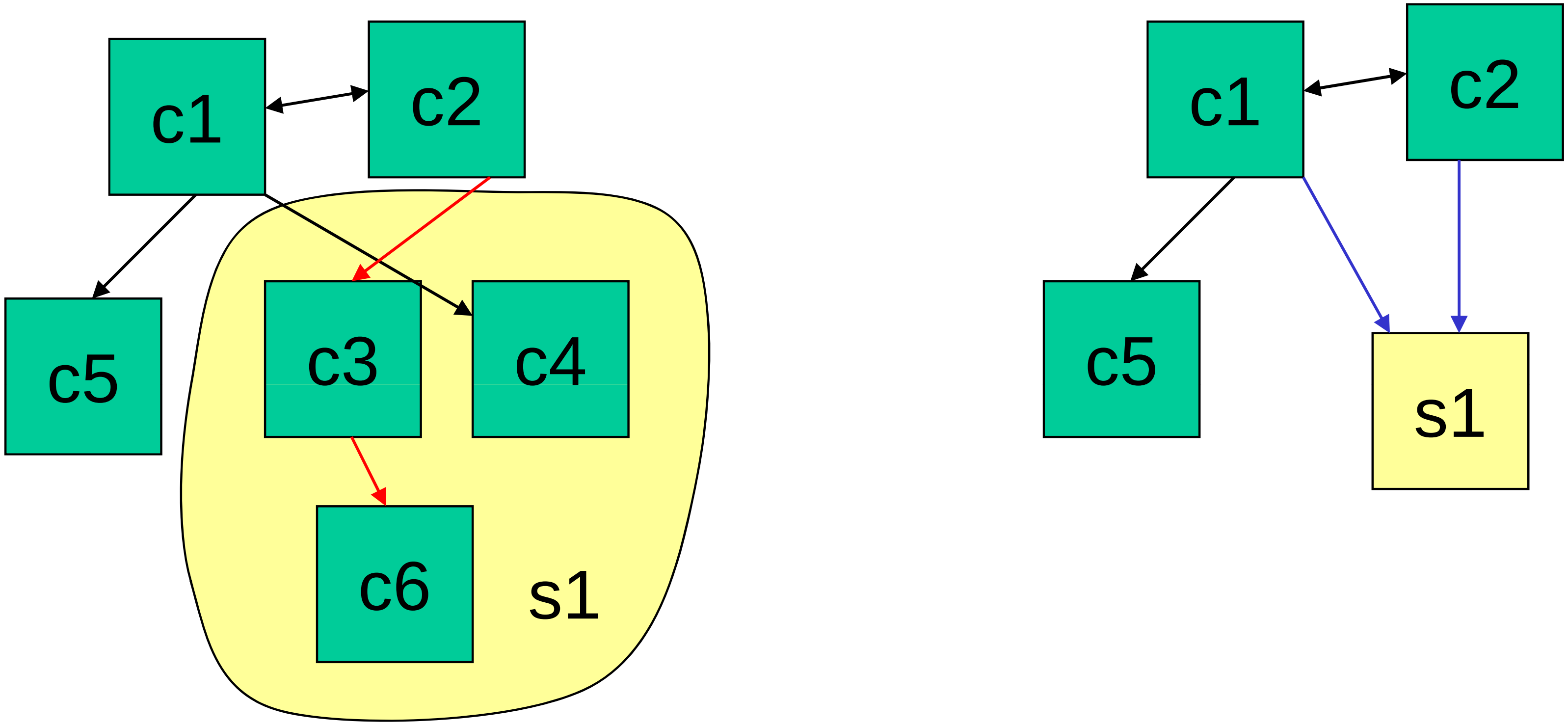
A simple picture explanation



Components interactions

- ✓ As a result of the systems’ partition into components, components interact with each other
- ✓ A first classification
  - ☞ There is direct action from component A to component B when A **can** invoke an access operation on B
  - ☞ There is proxy action from component A to component B when A **can** invoke an access operation on B by means of a direct or proxy interaction with a third component C
- ✓ Interaction: mutual action between components

Openness



c1 is open to c2; c2 is open to c1; c5 is open to c1;
c3 is open to c2; c6 is open to c3;s1 is open to c1 and c2

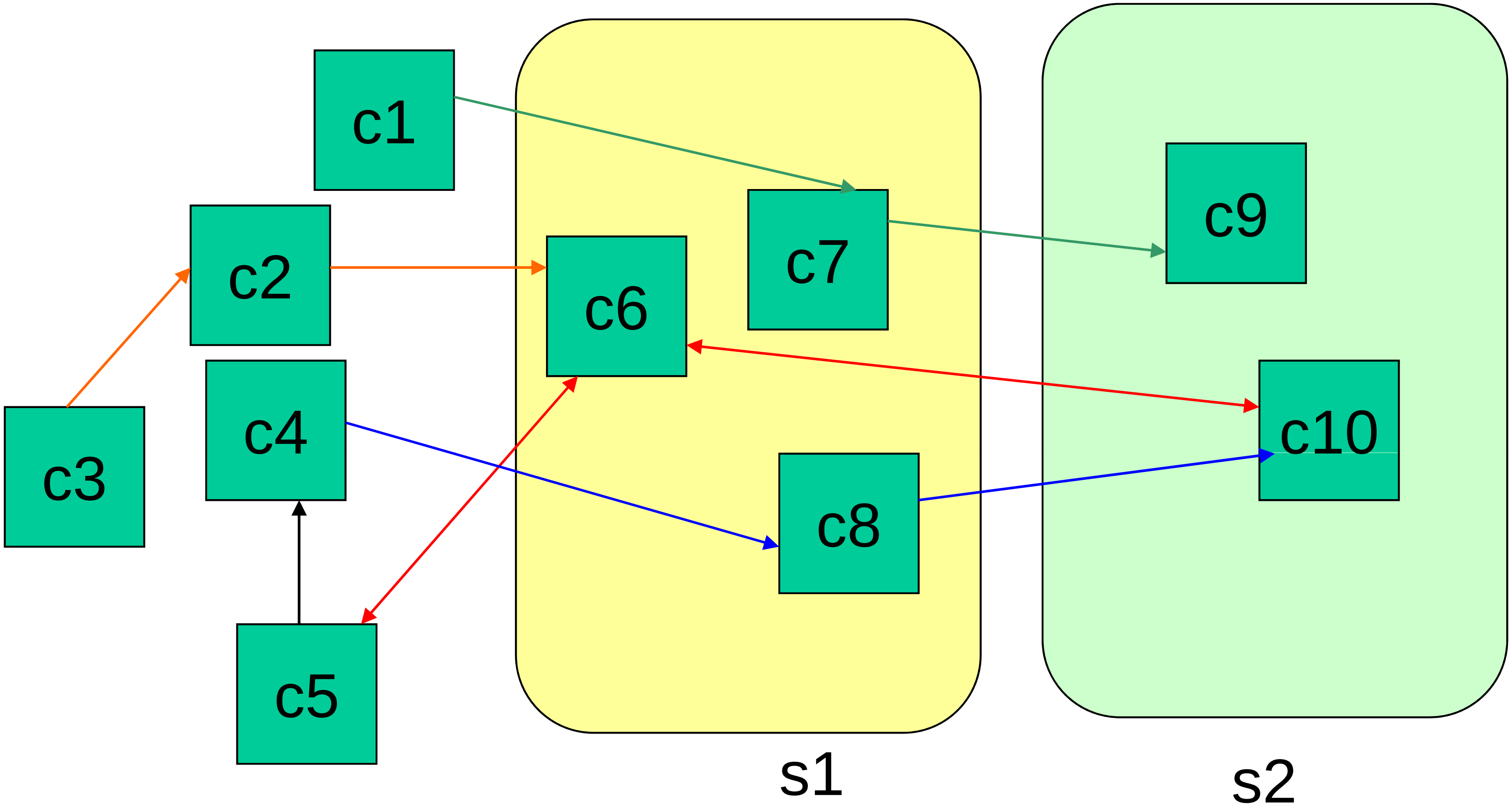
Open and Closed systems

- ✓ We can define a whole system **open** if it is accessible at some of its components,
  - ☞ i.e. if some of its components are accessible from external users and systems
- ✓ If all components are accessible to any user then the system is **absolutely** open

Layering

- ✓ A component A is a *layer* for another component B if
  - ☞ Every interaction between B and other components of the system different from A
    - ➢ is proxy
    - ➢ is done through a direct or proxy interaction with A

A simple picture explanation

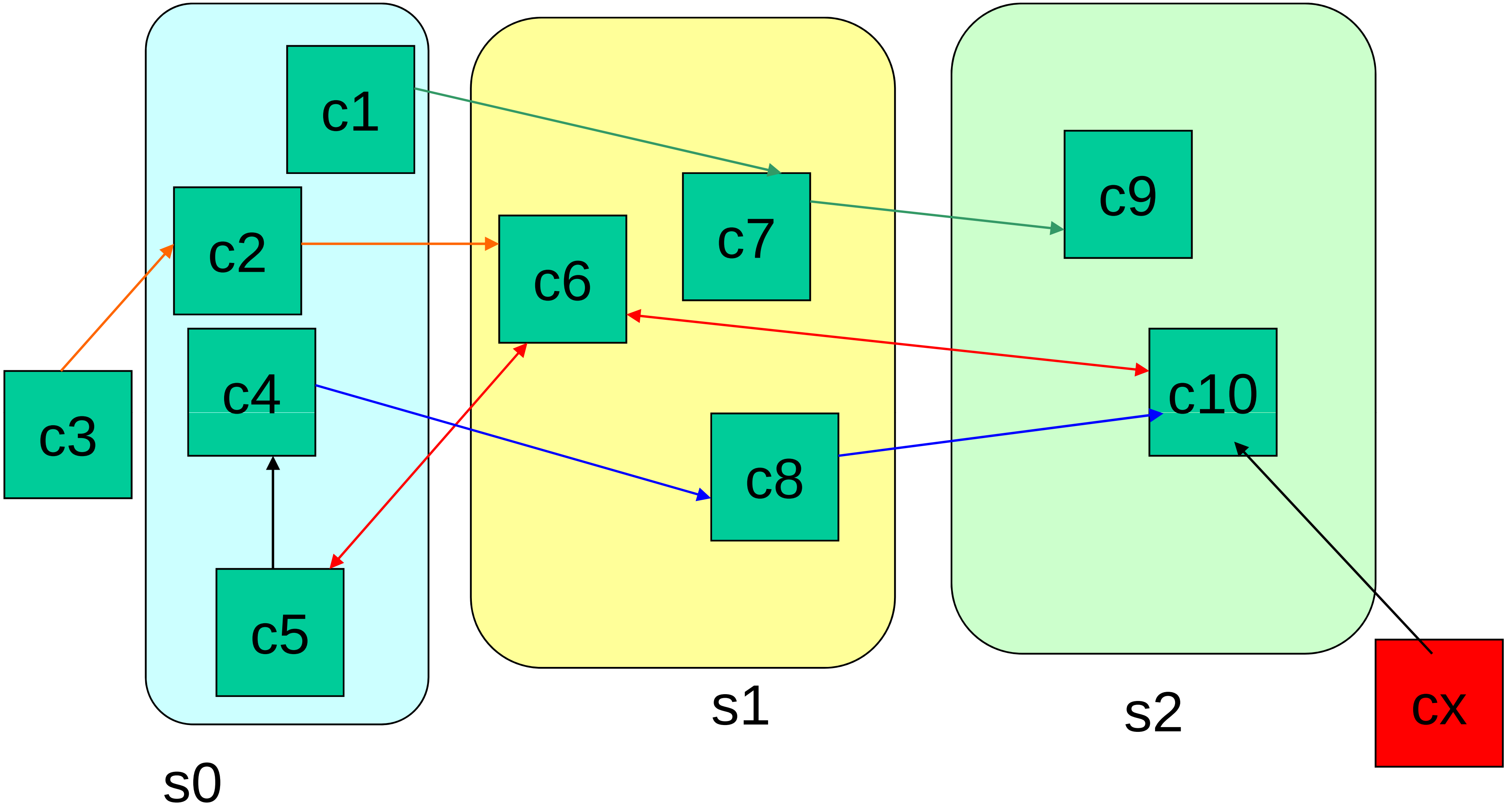


s1 is a layer for s2

Layered systems

- ✓ Definition: a system is ***layered*** if it presents a certain structure of layers, i. e. if their components are ordered in such a manner that one is a layer for the following
- ✓ A system is “absolutely layered” if this is true for every class of users

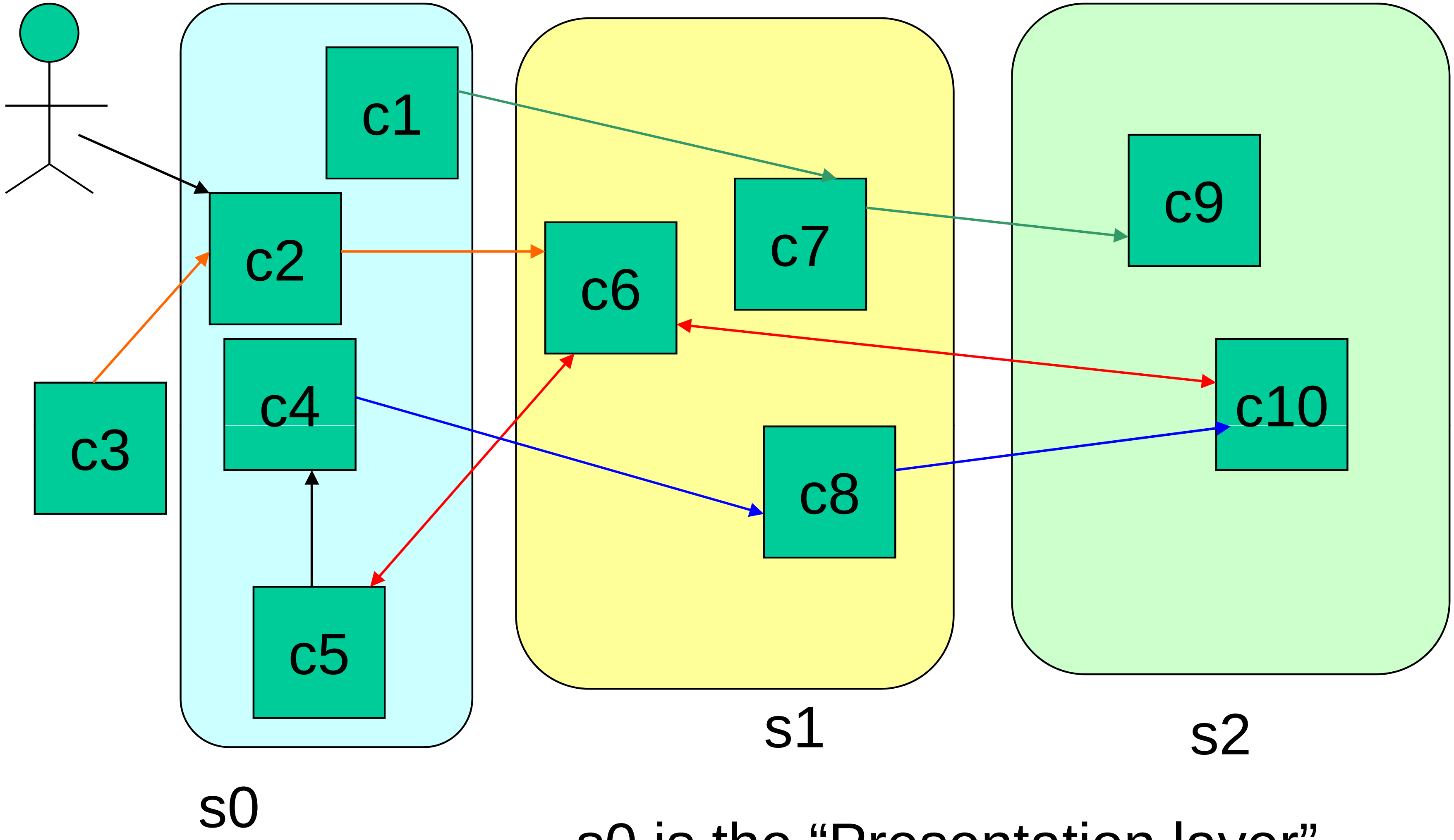
Layered and not layered



Component Roles

- ✓ Components having similar roles are generally aggregated to form subsystems
  - ☞ From the integrator’s perspective these could be for example
    - ➢ Application, services, storage
    - ➢ Application, data, storage
    - ➢ Presentation, application, storage
    - ➢ … (and various other permutations)
  - ☞ These are role names that components play in a system
- ✓ It’s important to notice that these classifications are not absolute: they are only helpful to describe the system from a determined perspective
  - ☞ Suggestion: remove this conceptual rigidity from the report, or at least clarify this aspect

Example



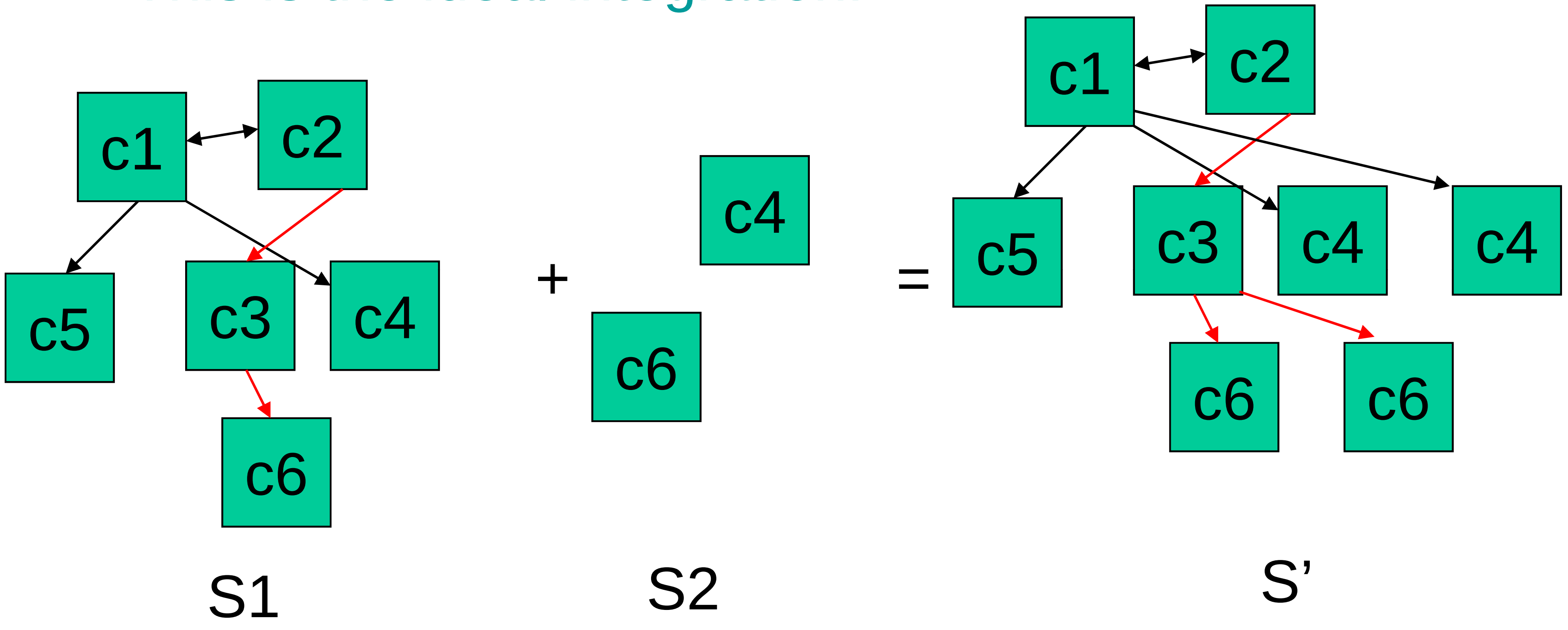
s0 is the “Presentation layer”

Integration

- ✓ Generally, there is integration when we want to produce a system combining a set of existing systems
- ✓ Different cases may arise from this process, exentially regarding the component structures of the resulting system
- ✓ In general the component structures of the combined system are different than those of the initial individual systems
- ✓ The user domain of the resulting system is the union of the user domains of the individual systems

Cases of Integration

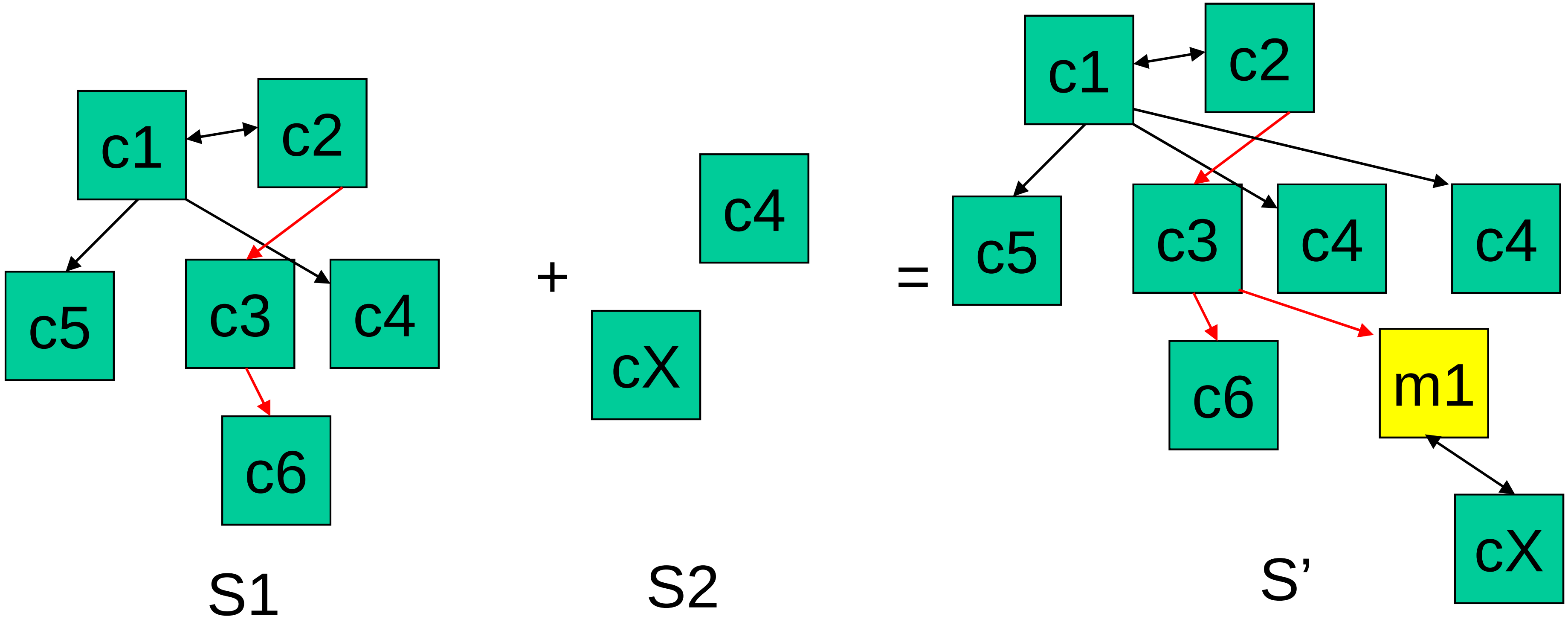
- ✓ Case 1: plug and play!!
  - ☞ This is the ideal integration!



Cases of Integration

- ✓ Is case 1 realistic?
  - ☞ Formally, real (not insignificant) integration cases always need some sort of middleware, because you have to adapt at least one of
    - ➢ Access methods & parameters, behaviour, semantics
- ✓ Case 2: extra components are needed
- ✓ Here is where the *middleware* components come out!
  - ☞ Their mission is in general to ***adapt*** the interactions between the components of the original systems
    - ➢ In terms of access methods and (I/O) parameters
    - ➢ In terms of semantics
    - ➢ In terms of behaviour

A simple picture explanation



Well, what can Middleware provide?

- ✓ Middleware are special components providing aids to solve the interoperability problems that raise from system integration
- ✓ This may stand for:
  - ☞ Structuring the systems by definition of interfaces.
  - ☞ Identifiying adaptation components between systems
  - ☞ Lowering of implementation efforts by availability of common services
- ✓ Know How Transfer from other industries.

What are services?

- ✓ Middleware components that are widely reusable in the integration processes often raise to the rank of “services”
  - ☞ So basically services can be defined as predefined components that are useful in system integration
  - ☞ Therefore, there is not a strict characterisation of services, except for their special role played in system integration
- ✓ From an implementation point of view
  - ☞ Services are software agents, providing functionalities which can be grouped due to their semantic affinity
  - ☞ Services need to specify how to use them
  - ☞ Services present one or several interfaces
- ✓ Examples: Control of realtime streams, Metadata management, Transfers, Authentication

Some (practical) suggestions

- ✓ Avoid sticking with stereotyped classifications (horizontal, vertical, open, closed, etc.) without giving a precise definition
- ✓ Systems characteristics are synthesis of their components'
  - ☞ Layered systems, open systems to be defined coherently with this aspect
- ✓ Don't confuse components' roles with their mission
  - ☞ Try not to use trilogies like "Application, Storage, Data" because they don't cover all interesting cases in broadcasting environment

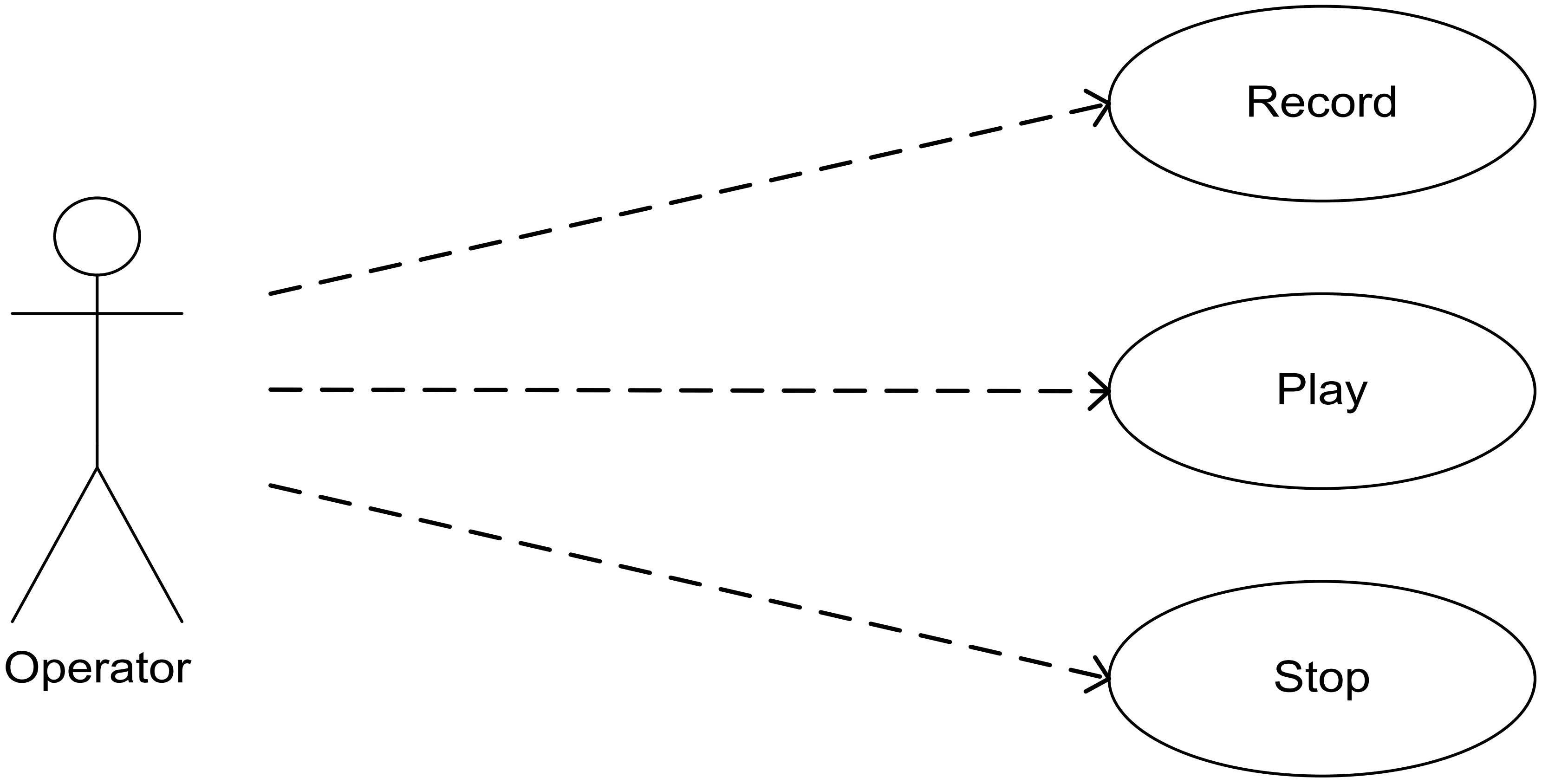
What are the enabling tools?

- ✓ Modelling is the process with which integration can be practically obtained
  - ☞ Identification of systems’ components
  - ☞ Identification of needed middleware components in the integration processes
  - ☞ Design of the middleware components
    - ➢ Identification, access methods and parameters, behaviour, semantics
- ✓ Modelling tools and philosophies give a wide range of facilities to do modelling
  - ☞ UML, MDA

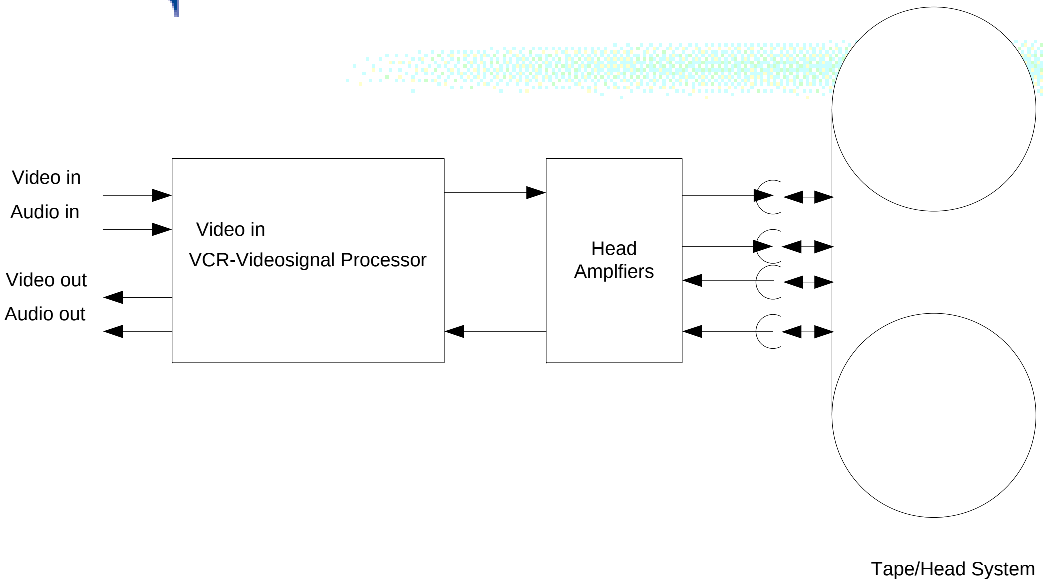
Modelling

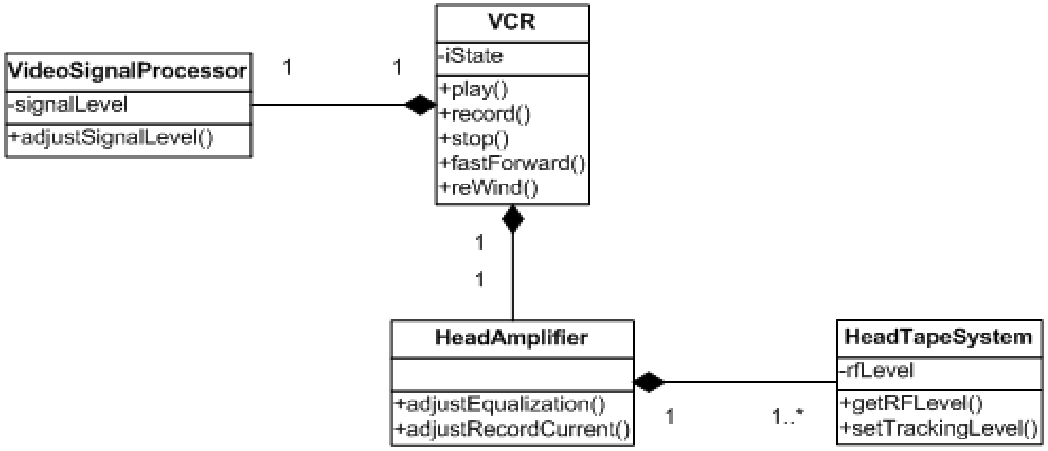
- ✓ Models have:
  - ☞ Different Views
    - ➢ Users, Designers, Implementers
  - ☞ Different Semantics
    - ➢ Different systems
  - ☞ Different Abstraction Level
    - ➢ Refinement process
  - ☞ Different Purpose
    - ➢ Design, integration
- ✓ UML – Unified Modelling Language is the prime technology for modelling

The Story Behind: UML – Use Case Analysis

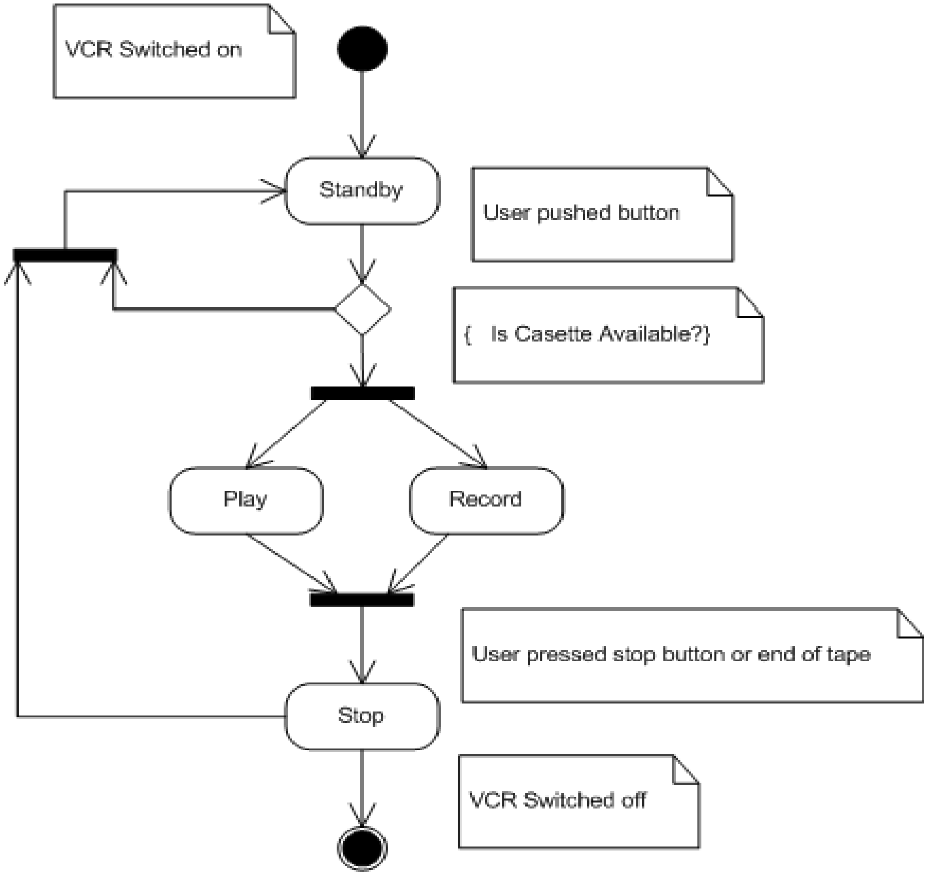


The Story Behind : UML - Static





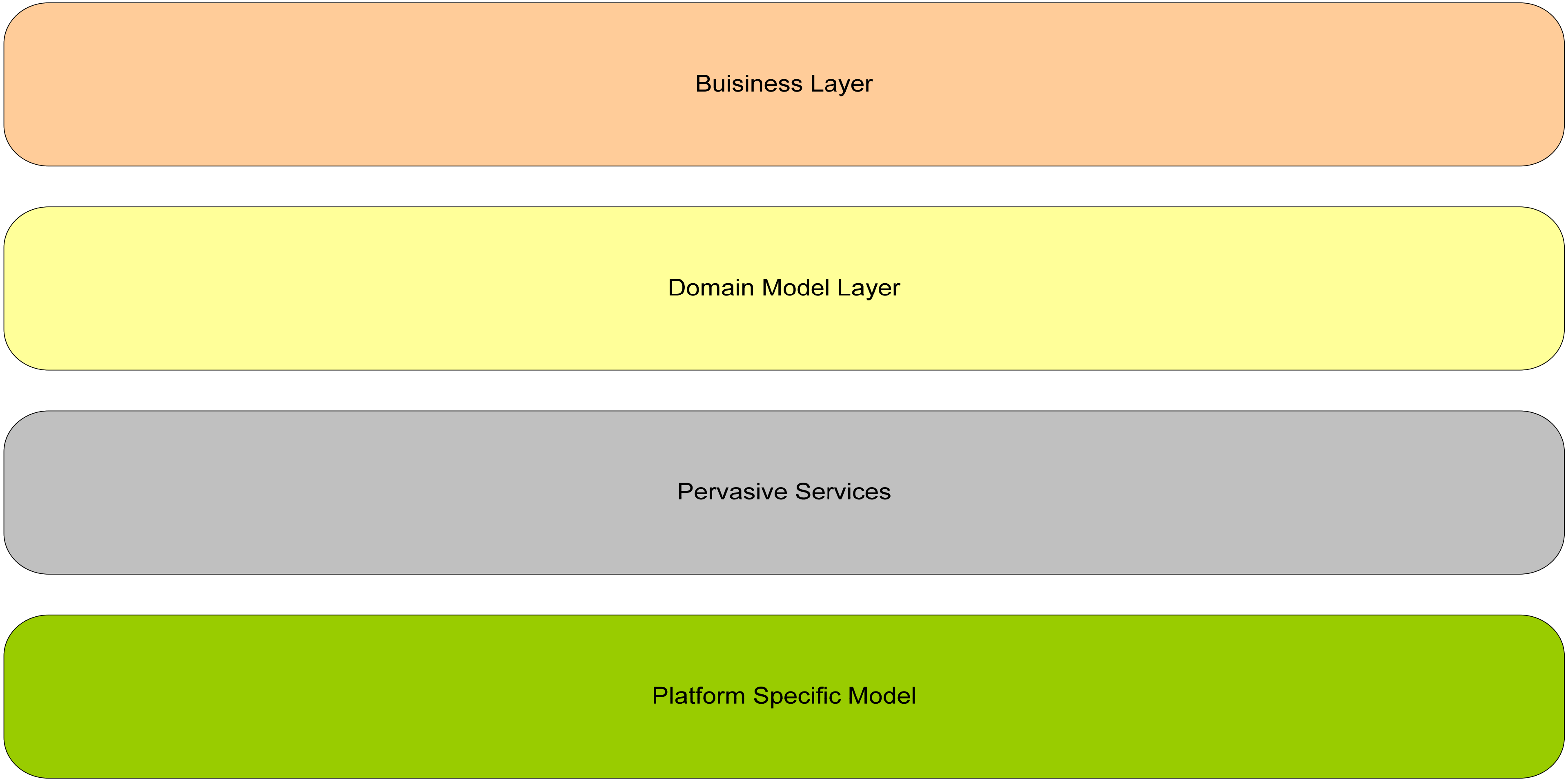
The Story Behind : UML - Dynamic



What is a typical roadmap?

- ✓ Specification of a Domain Model i.e. Platform independent Model.
- ✓ Definition of broadcast specific services
- ✓ Definition of interfaces
  - ☞ Components of Systems to be integrated
  - ☞ Components of Integrated system
- ✓ Identification of appropriate technologies for the implementation of the Services layer.

Layered model used as basis for work in P/MDP



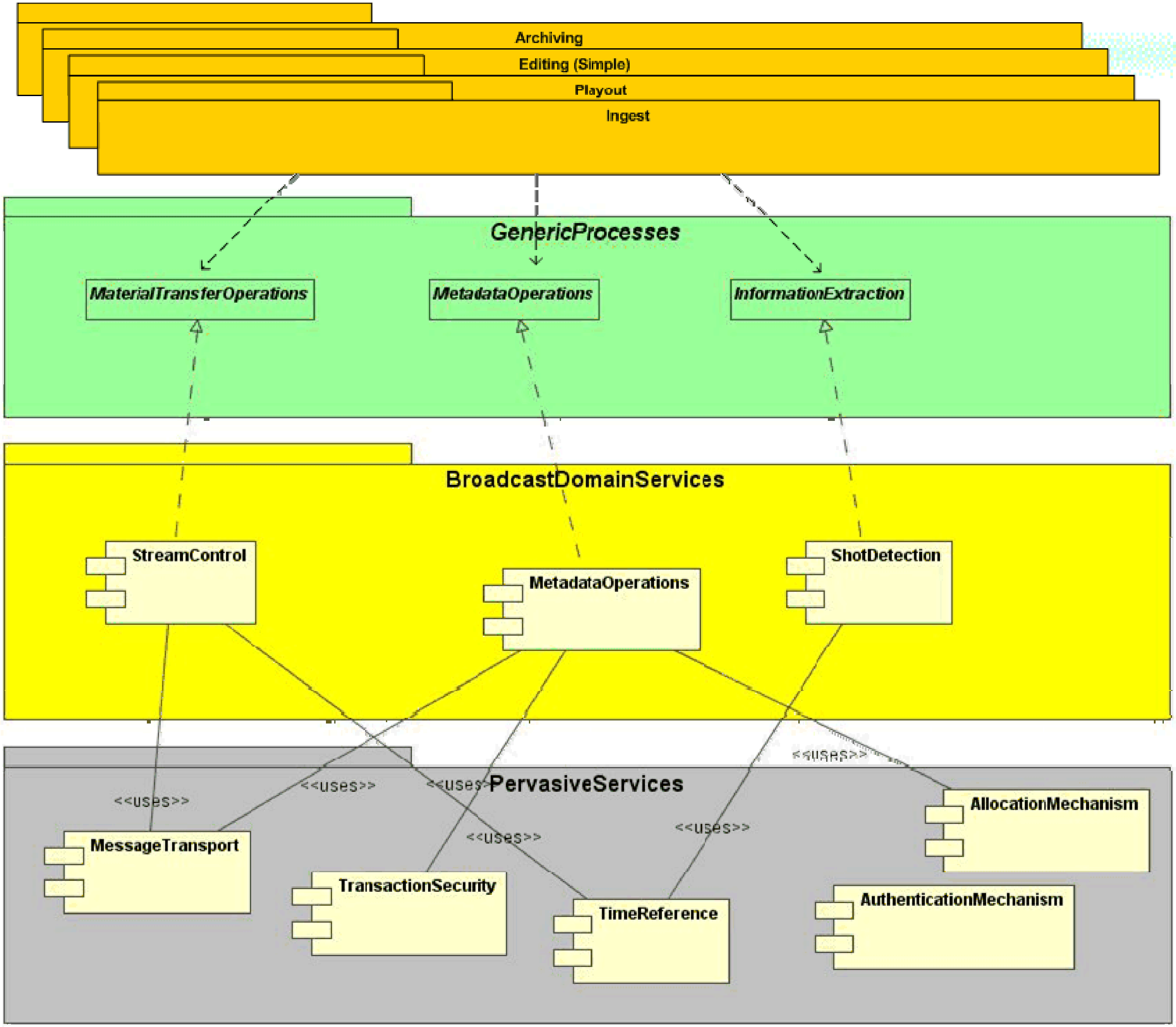
Buisiness Layer

Domain Model Layer

Pervasive Services

Platform Specific Model

Ingest case



This is
a layer →

What are Interfaces?

- ✓ From an implementation point of view the definition of an interface must include:
  - ☞ Message Format
  - ☞ Message Catalogue
  - ☞ Business Objects
  - ☞ Timeline behaviour
  - ☞ System behaviour

- ✓ E.g.:Play(Filename, StartTC, EndTC)

Conclusions

- ✓ Actually the interesting topic is System Integration
- ✓ Terminology is a key aspect
  - ☞ Middleware is a loose term denoting the components playing disparate roles in the system integration task
  - ☞ Services are special middleware components
- ✓ Don’t expect to see “comprehensive cheap easy-to-implement middleware-based solutions”
  - ☞ They simply **can’t** exist
- ✓ Modelling is a core activity in this context
  - ☞ Don’t be scared
  - ☞ Start studying!
- ✓ Effort in system integration can be lowered
  - ☞ Well specified interfaces for systems’ components

Authors and contributions

- ✓ Other authors
  - ☞ Hubert Hubbes, IRT, Germany
- ✓ Direct contributors
  - ☞ Steven Van Assche, VRT, Belgium
  - ☞ Wolfgang Englmaier, BR, Germany
- ✓ Acknowledgements
  - ☞ All the P/MDP members