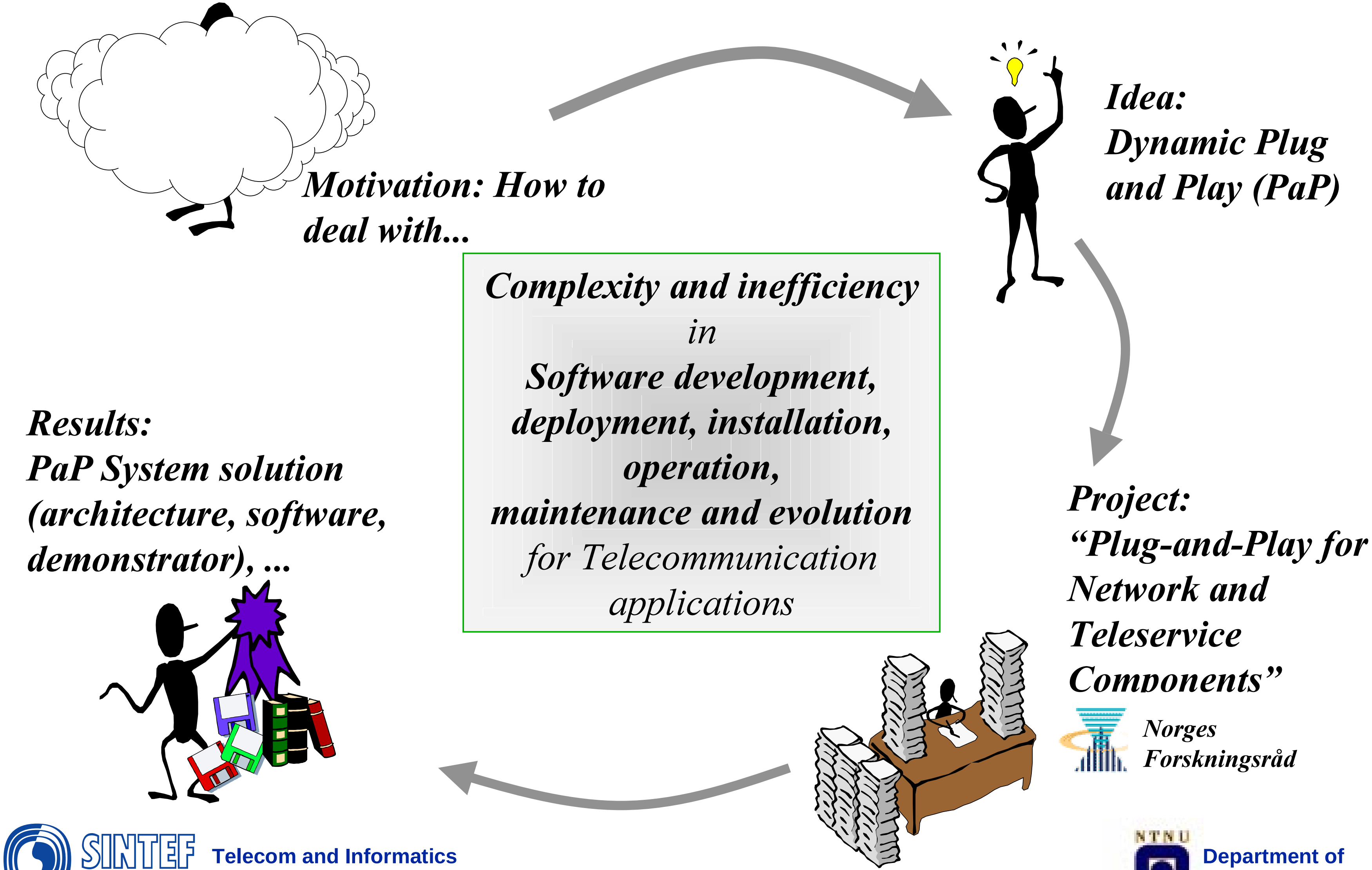
Dynamic Plug and Play (PaP)

What is it,
what are the advantages of using it?

Ulrik Johansen, SINTEF Tele og Data

Motivations, Idea, Project and Results



***Motivation:** How to deal with...*

Complexity and inefficiency
in
Software development, deployment, installation, operation, maintenance and evolution
for Telecommunication applications

Idea:
Dynamic Plug and Play (PaP)

Project:
"Plug-and-Play for Network and Teleservice Components"

Norges Forskningsråd

Results:
PaP System solution (architecture, software, demonstrator), ...

Definition of Static and Dynamic Plug-and-Play

PaP component

is some real-world reactive hardware or software with the capability of running Plug-and-Play functionality software.

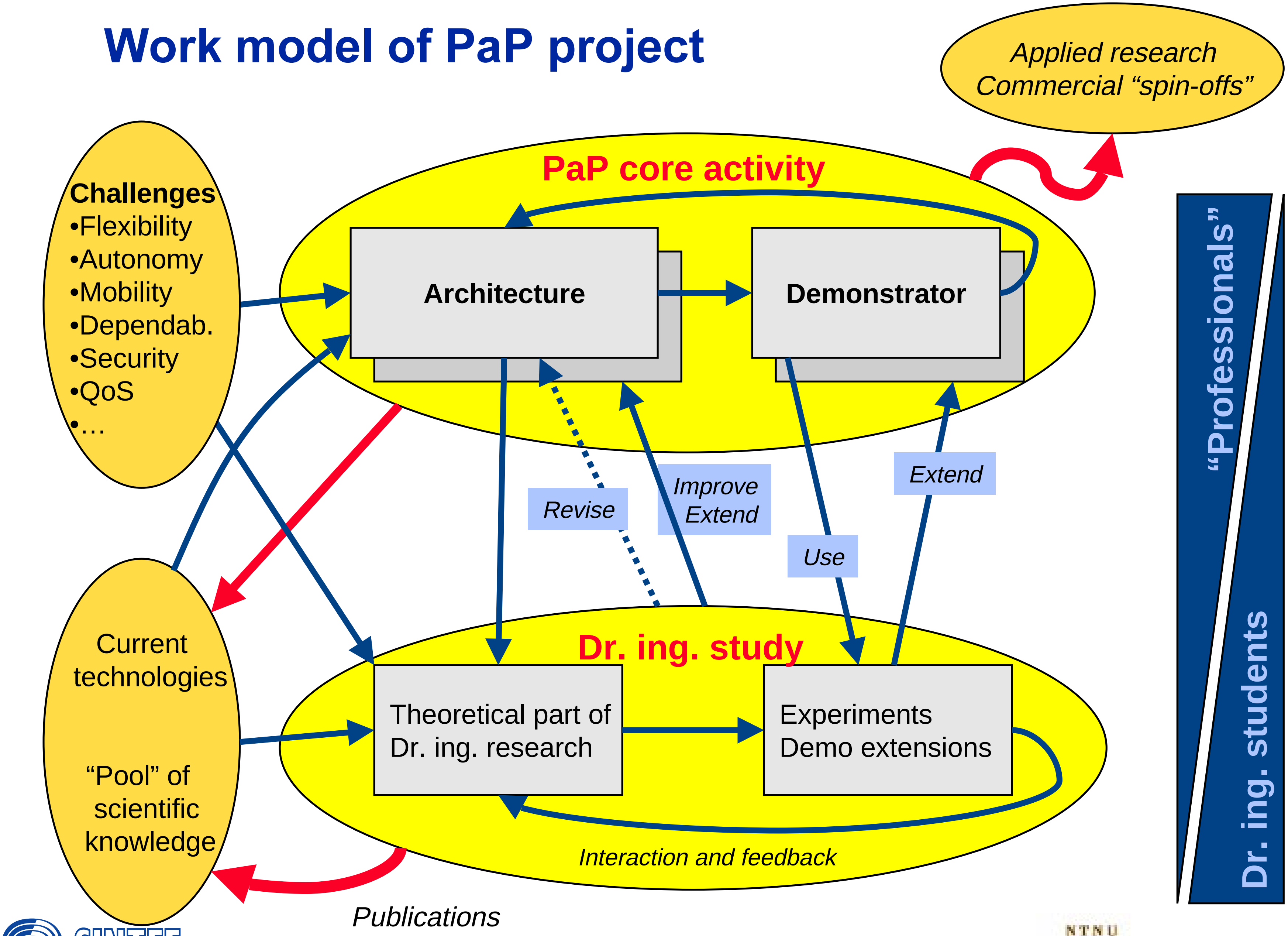
- **Static "Plug and Play"**

  Static Plug-and-Play components:
  - *Configures themselves at installation (to plug)*
  - *Provide services (to play) according to its predefined functionality*

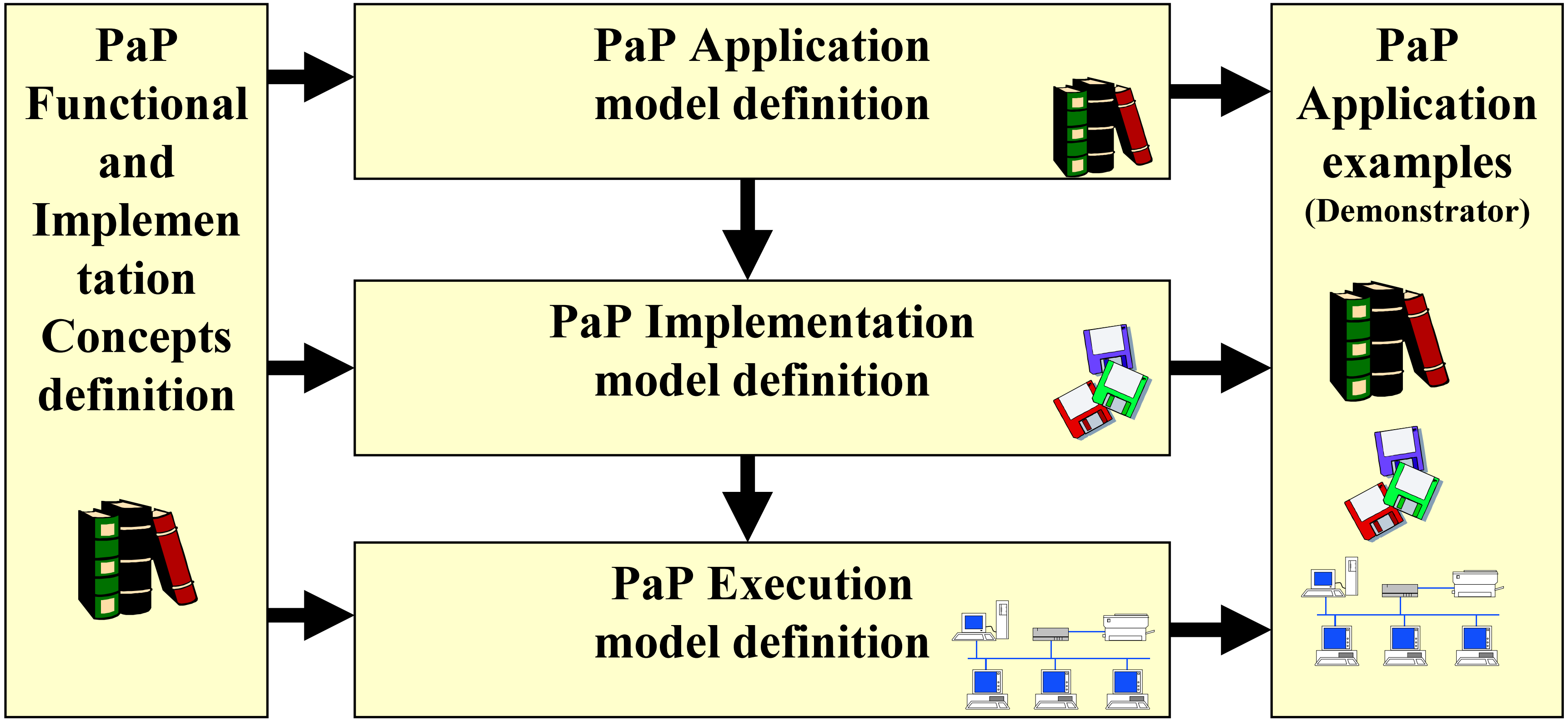
- **Dynamical "Plug and Play"**

  Dynamic Plug-and-Play components:
  - *Have a set of basic capabilities*
  - *Their functionality is decided during the plug-in procedure*
  - *Their functionality can dynamically be changed during the lifetime of the component*

Work model of PaP project



PaP core activity



PaP Functional and Implementation Concepts definition

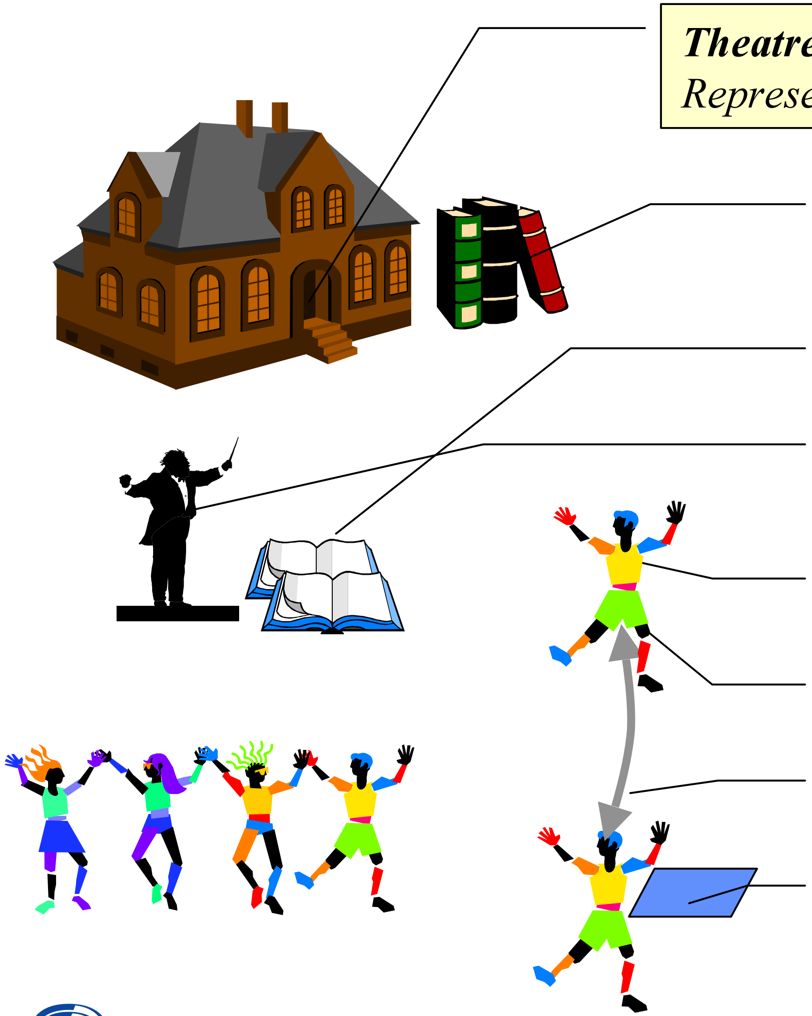
PaP Application model definition

PaP Implementation model definition

PaP Execution model definition

PaP Application examples (Demonstrator)

PaP Functional Concepts definition



Theatre: *A metaphor for concepts and functionality definition. Represents a* ***PaP Domain****.*

Repertoire: *The set of* ***Plays*** *that may be performed at the theatre.* ***PlayingBase*** *defines all actors.*

Play*: Defines a set of logically related functionality.*

Director*: The manager of plays, and supervisor for actors. Director is an actor.*

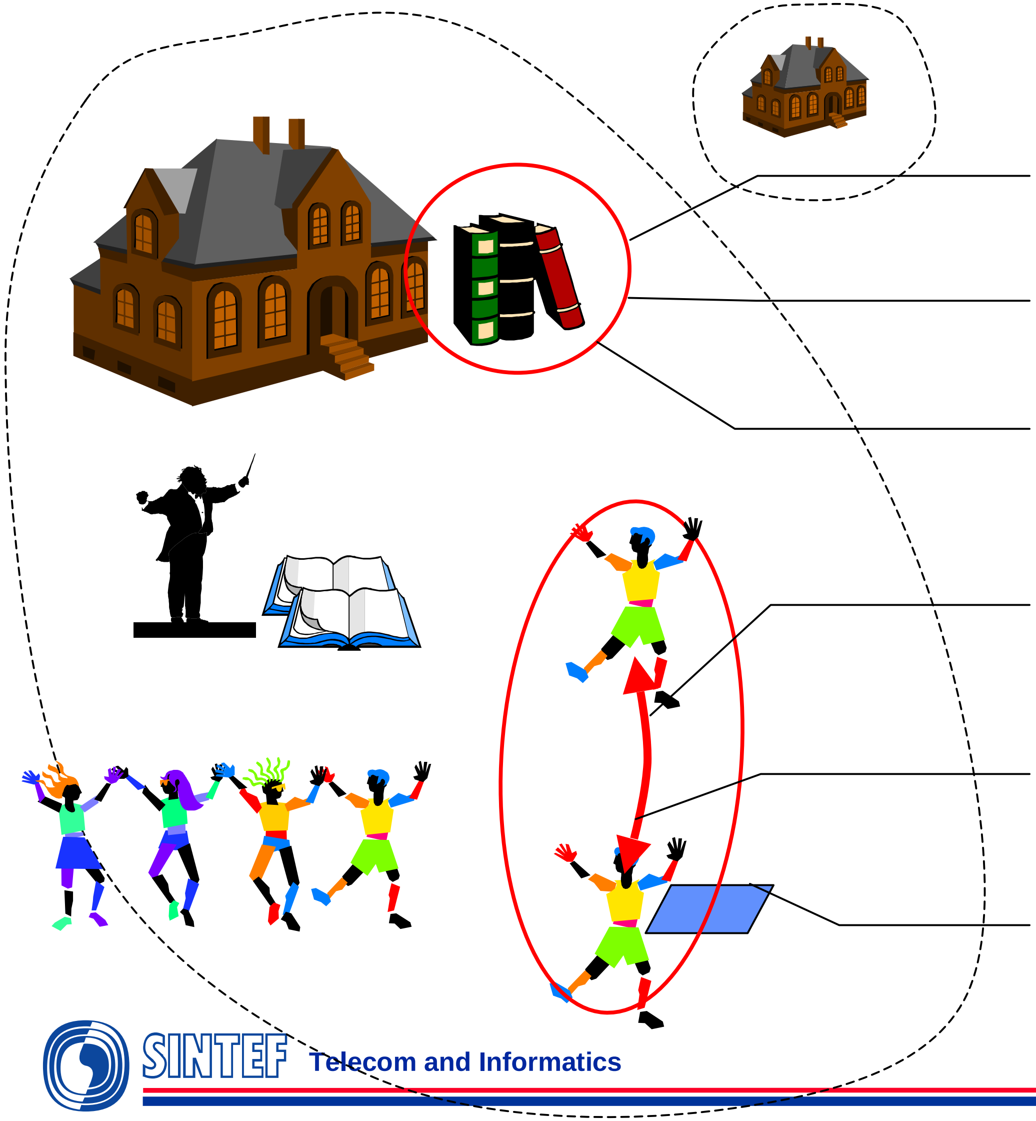
Actors*: The performers of plays. Plays a* ***Role****.*

Capability*: A unique set of properties of an actor.*

RoleSession*: A dialogue between two Actors.*

Manuscript*: The assigned behaviour for an actor.* ***RoleSessionCombinder*** *co-ordinates RoleSessions.*

Plug Functionality



***Context**: Play must have been defined*

***PlayPlugIn**: Make new functionality available as a new play version.*

***PlayChangesPlugIn**: Change functionality of an existing play version.*

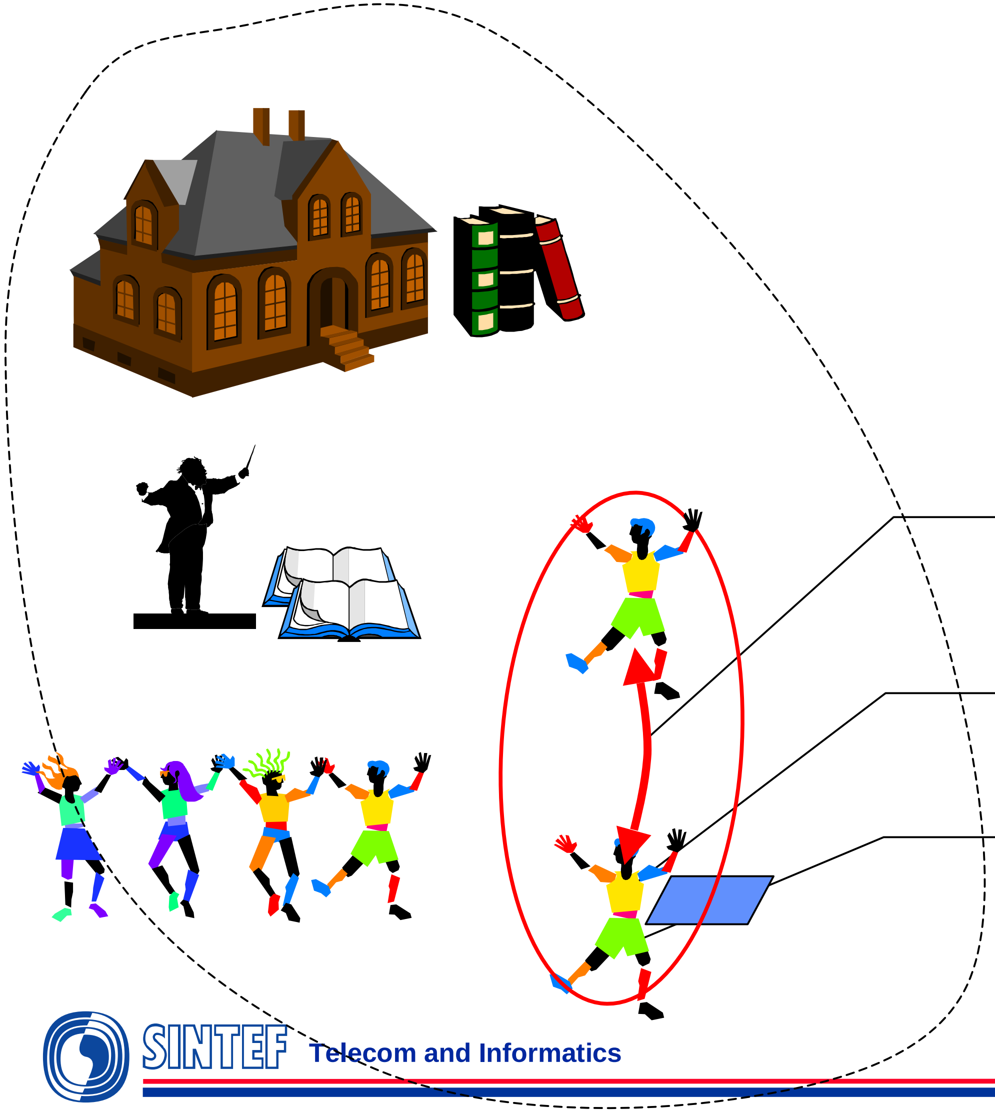
***PlayPlugOut**: Remove a play version*

***ActorPlugIn**: Establish a role session between two actor. An actor may be created implicitly.*

***ActorPlugOut**: Terminate a role session. An actor may implicitly be discarded.*

***ActorChangeBehaviour**: Replace actor behaviour by new manuscript plug-in.*

Play Functionality



Context*:*
Play plugged in
Actors plugged in

RoleSessionAction*: A message from one actor to another or to self*

ChangeActorCapabilities*: Change specific properties for own actor*

SubscribeEvents*: Request to be notified for specified events.*

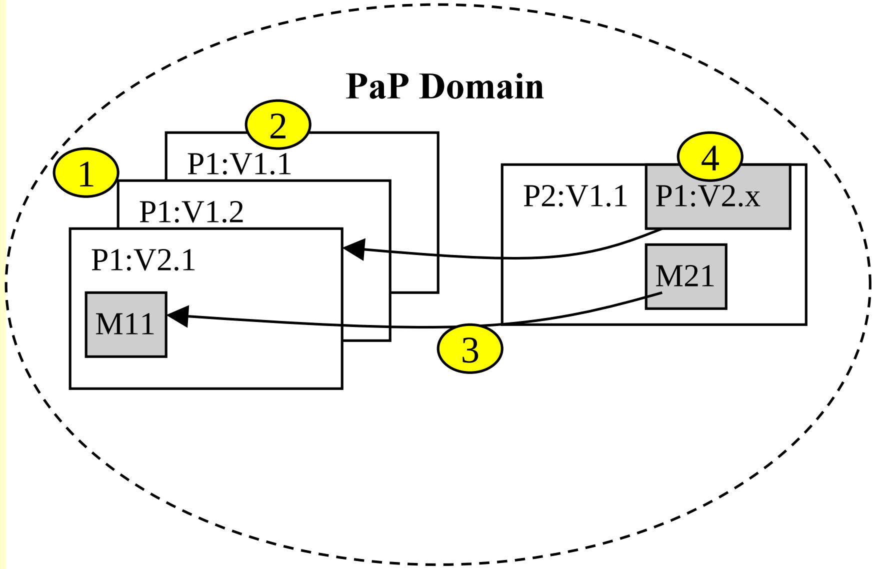
Properties related to Play-plug functionality

1 **Play versions** of the same play within same PaP domain is allowed

2 **Compatible/not compatible versions** of same play is explicit defined through major/minor ver.no.

3 **Plays explicit related** through use of ActorPlugIn in Manuscripts

4 **Compatibility between related plays** is explicit defined

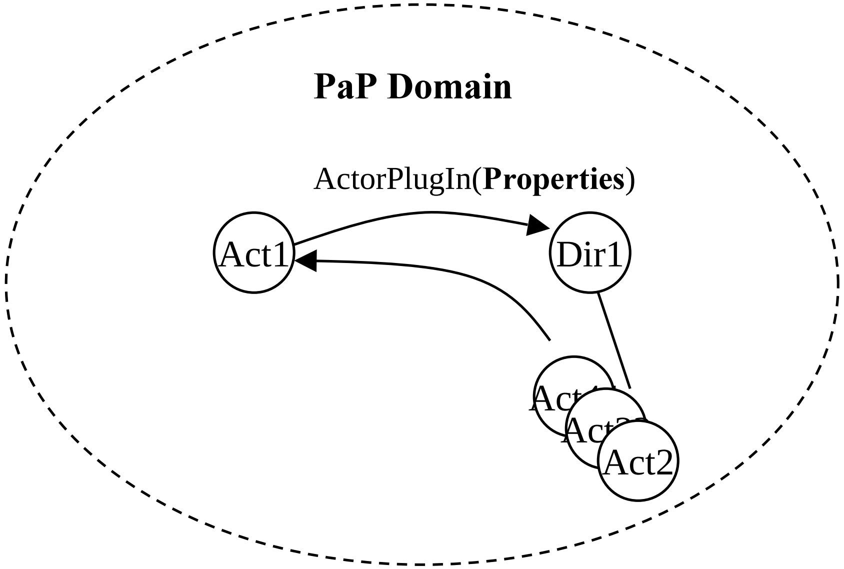


Advantages: Robustness and flexibility through

- **Functionality consistency assurance**
- **Dynamic update of functionality**

Properties related to Actor-plug functionality

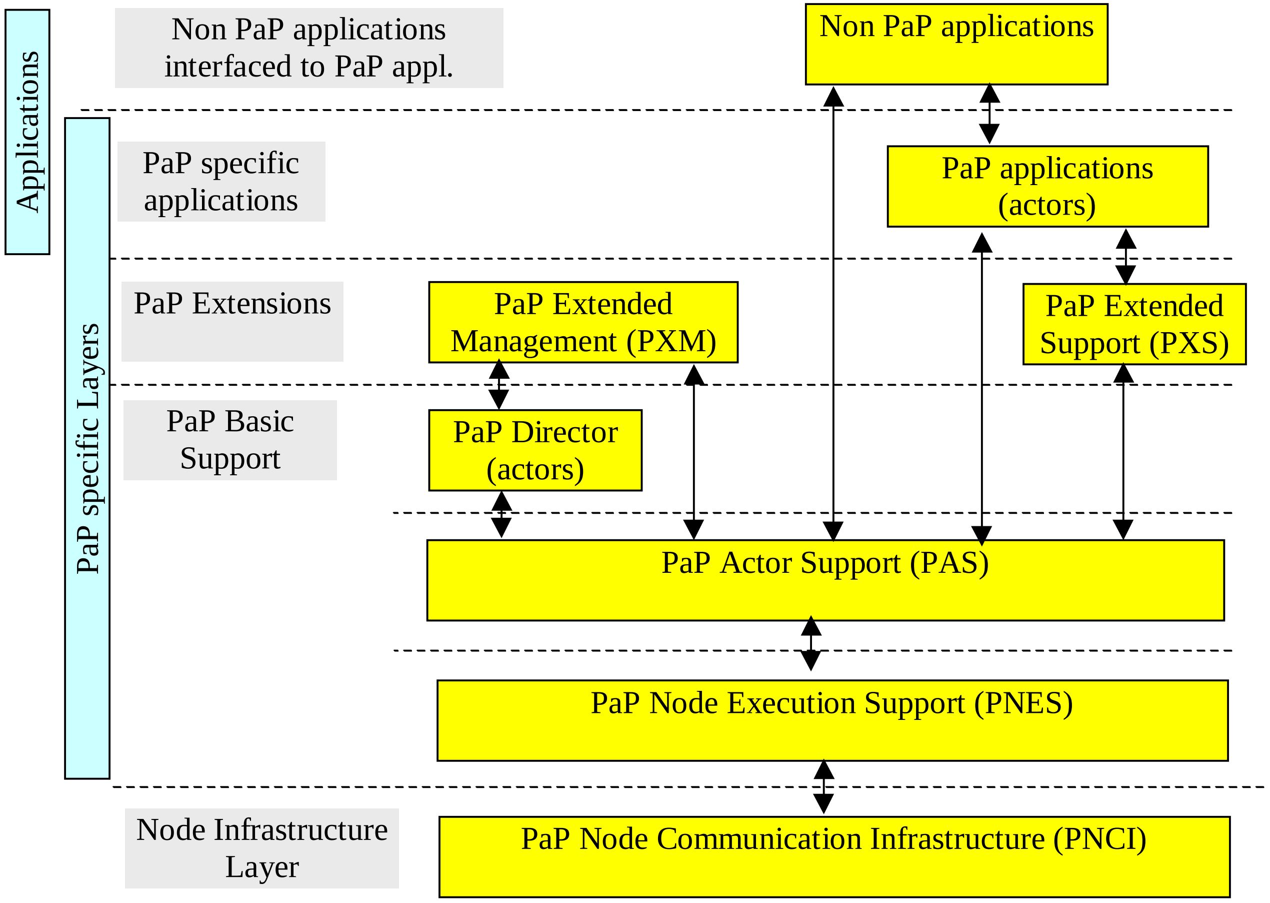
- Actor-plug specifies properties for an actor, not an actor
- Properties must include *role*, and optionally one or more of:
  - *Location, eventually a specific actor*
  - *Required/Requested Capabilities*
  - *Required/Requested QoS*
  - *Visibility*
  - *SelectStrategy*



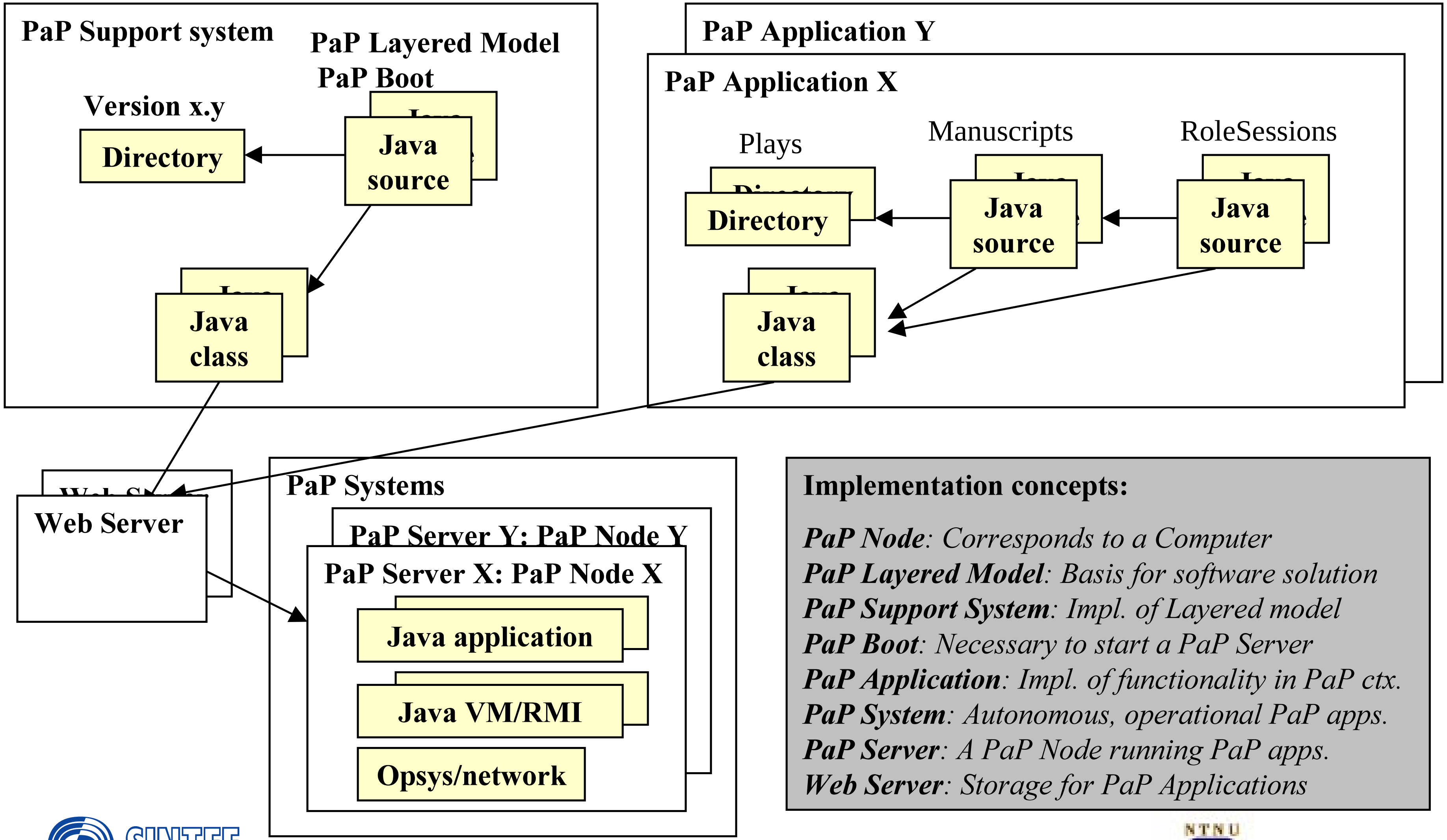
Advantages: Flexibility and adaptability through

- Request for functionality instead of specific actor
- Dynamic determination of operational architecture

The Layered Model


Applications
PaP specific Layers
Non PaP applications interfaced to PaP appl.
PaP specific applications
PaP Extensions
PaP Basic Support
Node Infrastructure Layer
Non PaP applications
PaP applications (actors)
PaP Extended Management (PXM)
PaP Extended Support (PXS)
PaP Director (actors)
PaP Actor Support (PAS)
PaP Node Execution Support (PNES)
PaP Node Communication Infrastructure (PNCI)

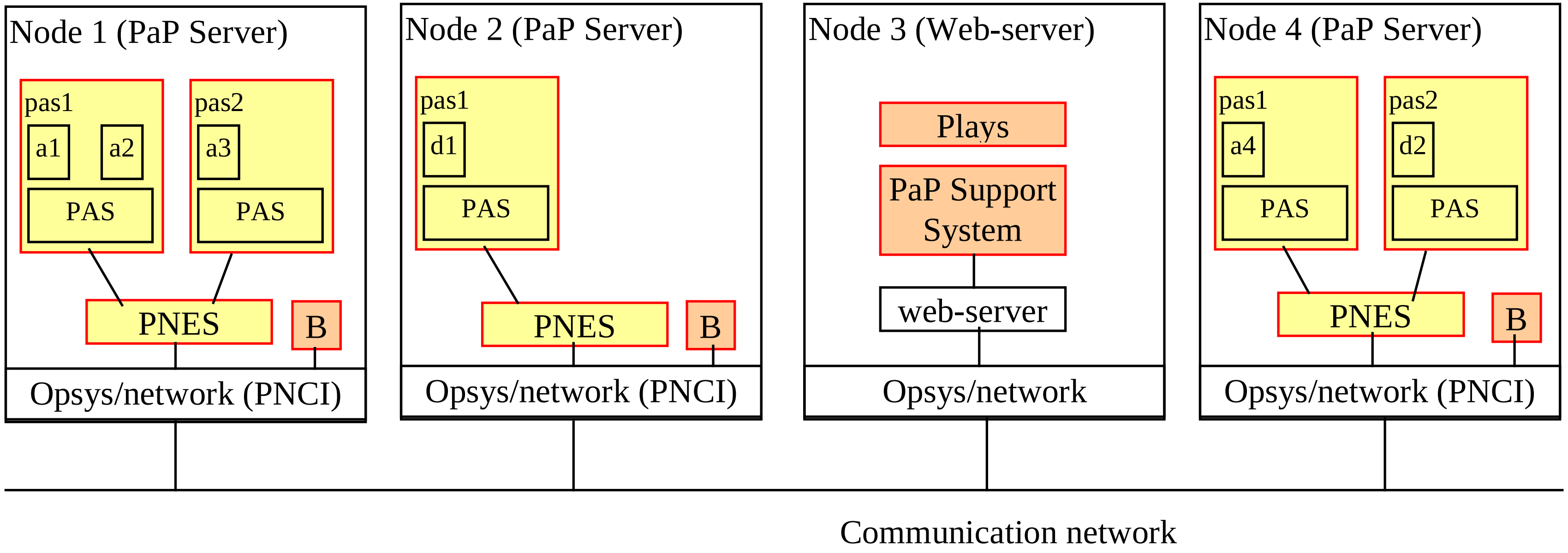
PaP Implementation concepts



Implementation concepts:

PaP Node: *Corresponds to a Computer*
PaP Layered Model: *Basis for software solution*
PaP Support System: *Impl. of Layered model*
PaP Boot: *Necessary to start a PaP Server*
PaP Application: *Impl. of functionality in PaP ctx.*
PaP System: *Autonomous, operational PaP apps.*
PaP Server: *A PaP Node running PaP apps.*
Web Server: *Storage for PaP Applications*

PaP System example



Node 1 (PaP Server)

pas1 a1 a2 PAS

pas2 a3 PAS

PNES B

Opsys/network (PNCI)

Node 2 (PaP Server)

pas1 d1 PAS

PNES B

Opsys/network (PNCI)

Node 3 (Web-server)

Plays

PaP Support System

web-server

Opsys/network

Node 4 (PaP Server)

pas1 a4 PAS

pas2 d2 PAS

PNES B

Opsys/network (PNCI)

Communication network

Legend:
a1 - a4: actor1 - actor4
d1, d2: director1, director2
B: PaP Boot

Legend:
🟧 Static available
🟨 Dynamic available

PaP Applications examples

- **”Tele-School” - A Network based learning application**
  Defines one play, four roles, four role sessions, and also interfaces to one Non PaP application

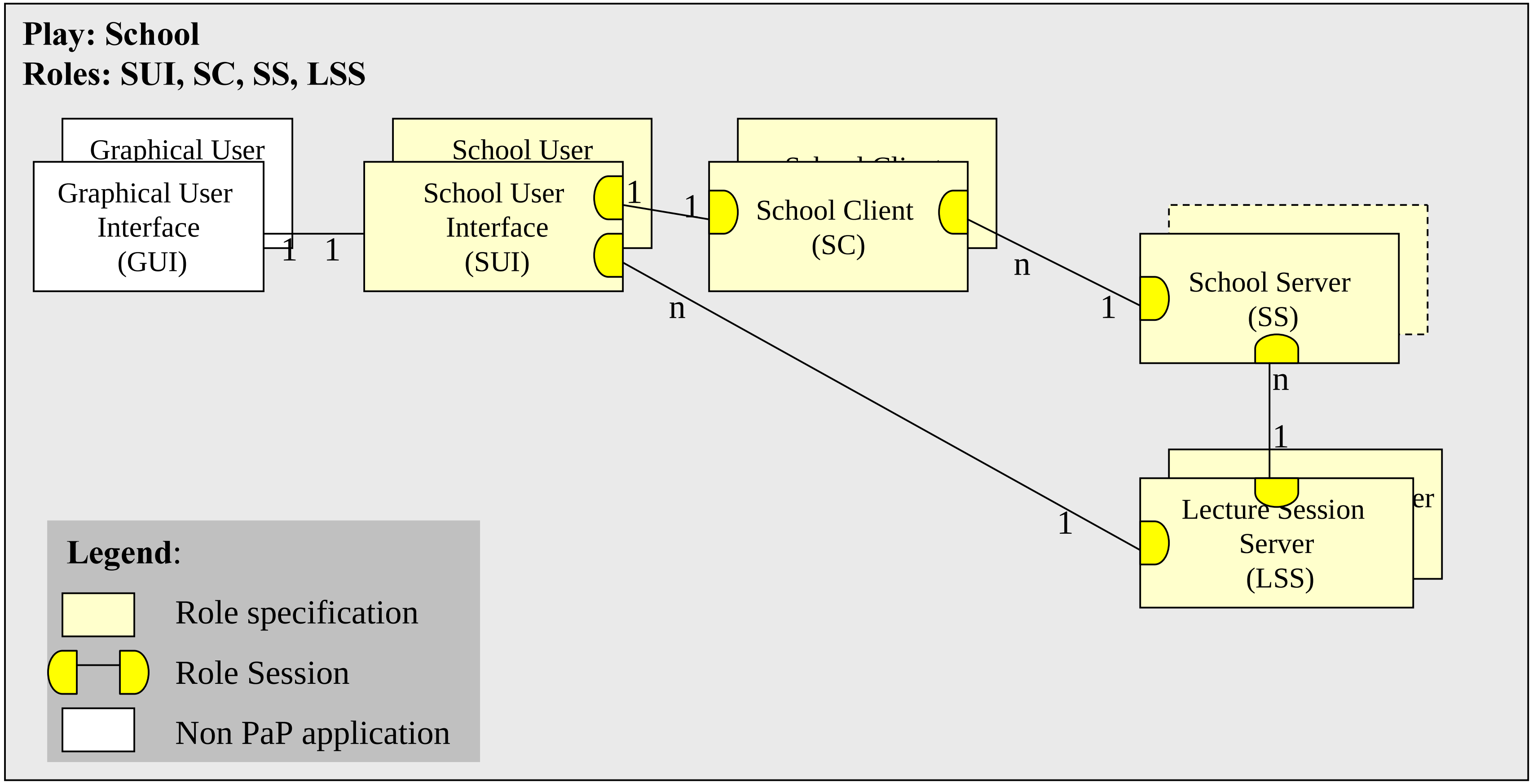
- **”Watcher” - A PaP Support activity monitor**
  Defines one play, one role and interfaces to one Non PaP application.

- **”TestPaP” - A tool for automatic testing of PaP Support**
  Defines three plays, one role for each play, six different role sessions, and interfaces to two Non PaP applications. Make use of different play versions.

“Tele-School” requirements

- A Network based learning application
- Main functional terms are:
  - *Teacher, Student, School, Course, Lecture*
- Main implementation terms are:
  - *Distributed solution, Mobility, Multimedia, Chat, Mail, News.*
- Main functions are:
  - *Real-time interactive lecture session (partly implemented)*
  - *Off-line learning session (not implemented)*
  - *Off-line teacher support to students (not implemented)*

“Tele-School” modelling



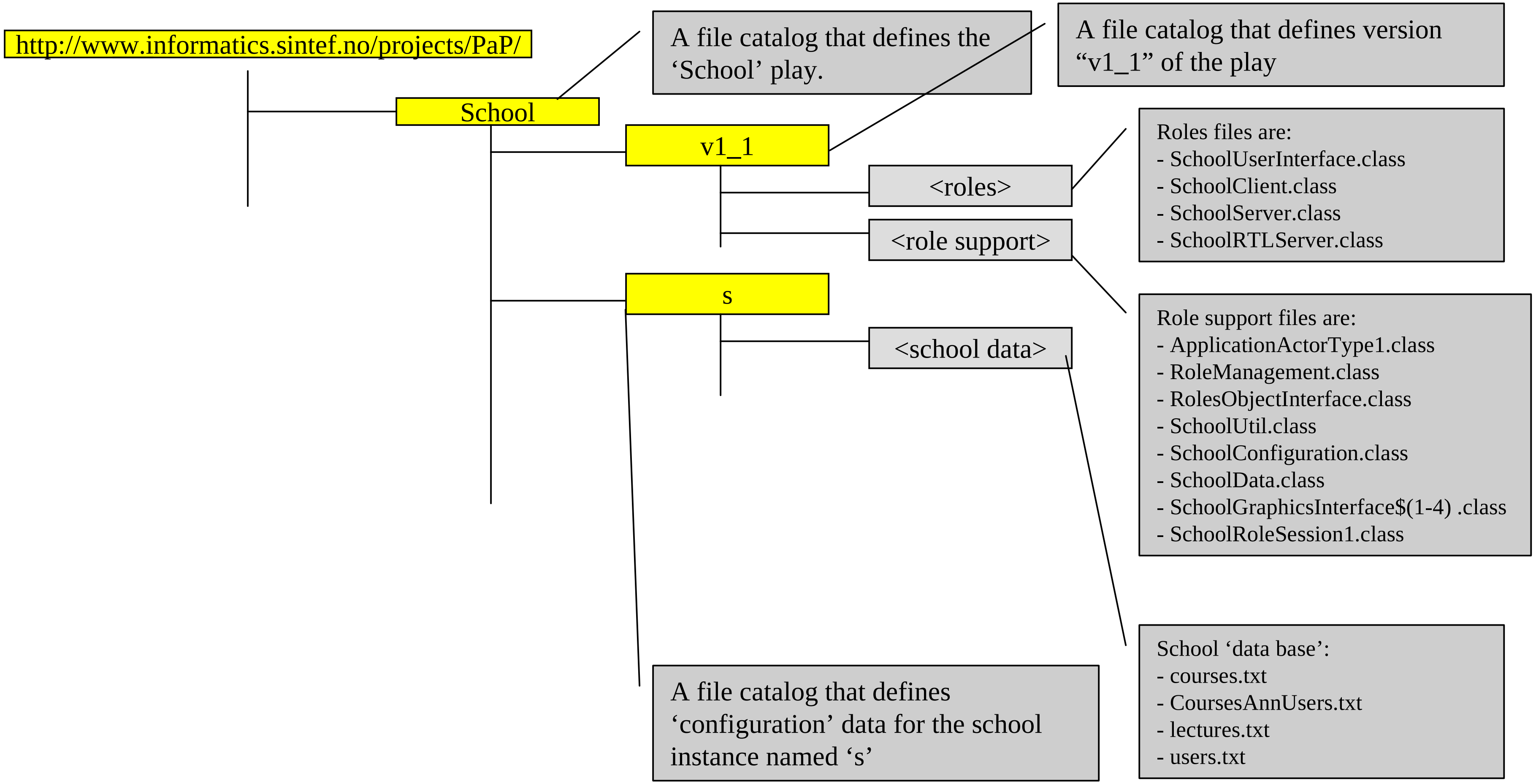
Play: School
Roles: SUI, SC, SS, LSS

Graphical User
Graphical User Interface (GUI)
School User
School User Interface (SUI)
School Client (SC)
School Server (SS)
Lecture Session Server (LSS)
er
1
1
1
1
n
n
1
n
1
1

Legend:

- 🟨 Role specification
- Role Session
- □ Non PaP application

“Tele-School” Implementation



http://www.informatics.sintef.no/projects/PaP/

School

v1_1

<roles>

<role support>

s

<school data>

A file catalog that defines the ‘School’ play.

A file catalog that defines version “v1_1” of the play

Roles files are:
- SchoolUserInterface.class
- SchoolClient.class
- SchoolServer.class
- SchoolRTLServer.class

Role support files are:
- ApplicationActorType1.class
- RoleManagement.class
- RolesObjectInterface.class
- SchoolUtil.class
- SchoolConfiguration.class
- SchoolData.class
- SchoolGraphicsInterface$(1-4) .class
- SchoolRoleSession1.class

School ‘data base’:
- courses.txt
- CoursesAnnUsers.txt
- lectures.txt
- users.txt

A file catalog that defines ‘configuration’ data for the school instance named ‘s’

Advantages of using PaP (1 of 3)

Development of PaP Applications

- **Flexibility in application modelling**
  “Composition” of Plays and Manuscripts from RoleSessions and RoleSessionCombiner
- **Transparency in distributed solutions**
  Use of Java/RMI
- **Portable**
  Use of Java
- **Mobile agents become possible**
  Uniform operational context for Actors, Java
- **Easy monitoring and controlling**
  Almost all PaP function requests served by Director

Deployment and Installation

- **Easy installation and maintenance of installations**
  Web-server, Play-plug functionality

Advantages of using PaP (2 of 3)

Operation

- **Dynamic change of behaviour at runtime**
  Use of Play-plug- and ActorChangeBehaviour- functions
- **Collaborative applications, in addition to client/server solutions**
  RoleSessions and RoleSessionAction function
- **Uniform execution environment for applications**
  PaP Actor Support as common context
- **Functional consistency assurance at runtime**
  RepertoireBase, Play versioning, PlayingBase
- **Security**
  Standardised operational environments for applications. All PaP communication routing is known.
  Utilisation of Operating system and Java security mechanisms

Advantages of using PaP (3 of 3)

Maintenance and Evolution

- **Software modification and extension**
  Play versions, Manuscripts and RoleSession definitions
- **Compact solution**
  requires only PaP Support System (50 classes, 120kb) and JRE^{TM}, in addition to the PaP application
- **Executable software and documentation available at Web:**

http://www.item.ntnu.no/~plugandplay

Concluding remarks

- Concepts and software that has potential to improve software development, deployment, installation, operation, maintenance and evolution.
- PaP solutions is based on available and portable technology (Java).
- Light weight solution for distributed, asynchronous, message based “soft” real-time applications.
- Ongoing improvements related to application modelling, fault tolerance, mobility and security (Dr.ing. and diploma work)