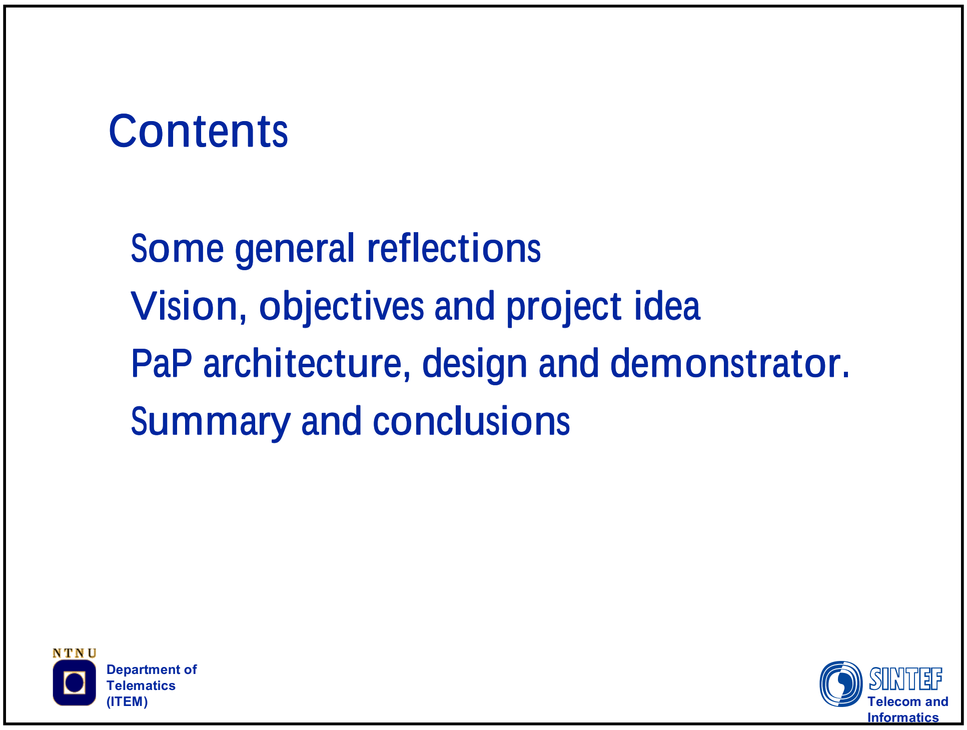
IConIT2001, May 28-31, 2001

Plug and Play (PaP) for Telecommunications - Architecture and Demonstration Issues.

Finn Arve Aagesen
Norwegian University of Science and Technology

An extended version of this presentation is available at:
http://www.item.ntnu.no/~plugandplay/IConIT.pdf



Contents

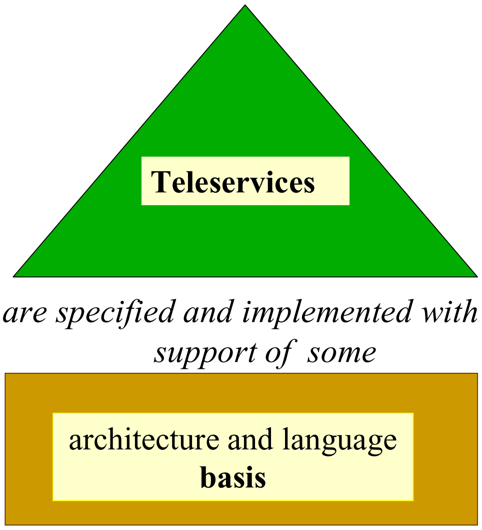
Some general reflections
Vision, objectives and project idea
PaP architecture, design and demonstrator.
Summary and conclusions

The grade of network intelligence:

"the *efficient flexibility* in the introduction of new teleservices and the *efficient flexibility* in the execution of teleservices"

<u>Teleservice examples</u>: CS-telephony, IP-telephony, Intelligent Network (IN)-services, Web-services, E-mail, FTP, Mini-banks and admission control services, Video conferencing, Tele-school, Nomadic office, Cooperative work, Tele-medicine, E-commerce, Remote sensing and control, Digital TV.

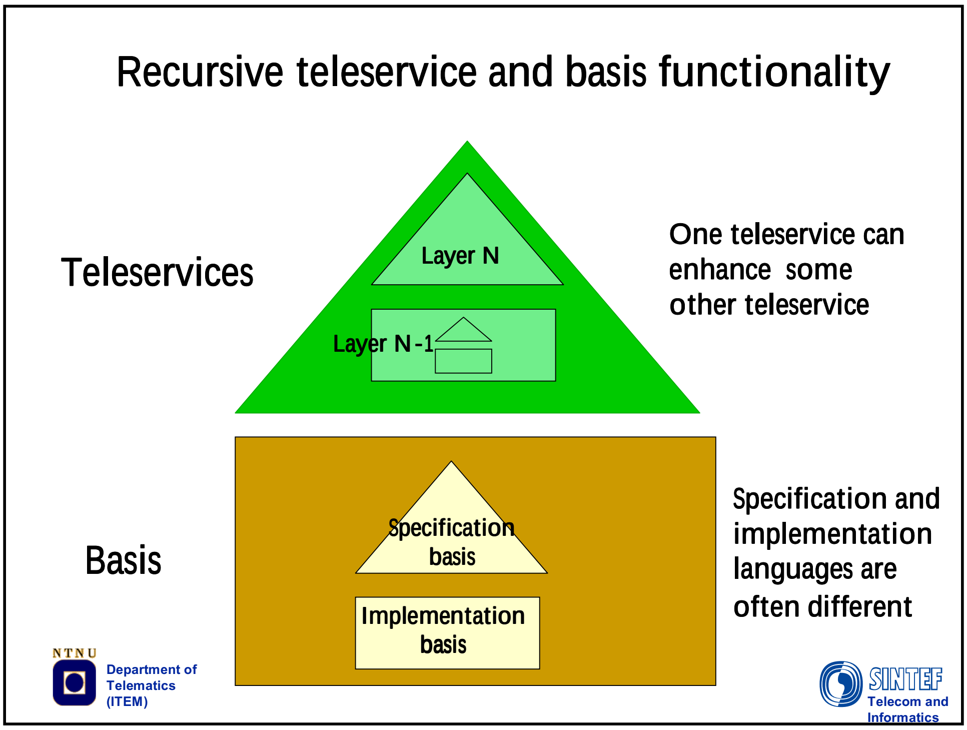
The creation of **substance** needs some **basis**



Teleservices

are specified and implemented with the support of some

architecture and language **basis**



Recursive teleservice and basis functionality

Teleservices

Layer N

Layer N-1

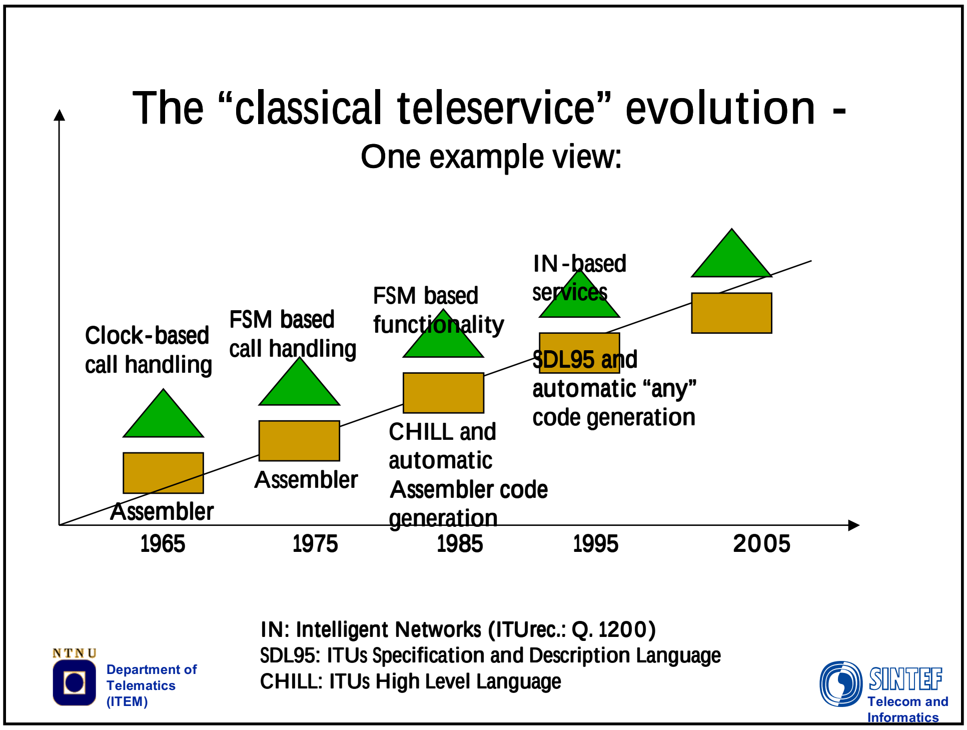
One teleservice can enhance some other teleservice

Basis

Specification basis

Implementation basis

Specification and implementation languages are often different



The “classical teleservice” evolution - One example view:

Clock-based call handling

Assembler

FSM based call handling

Assembler

FSM based functionality

CHILL and automatic Assembler code generation

IN-based services

SDL95 and automatic “any” code generation

1965

1975

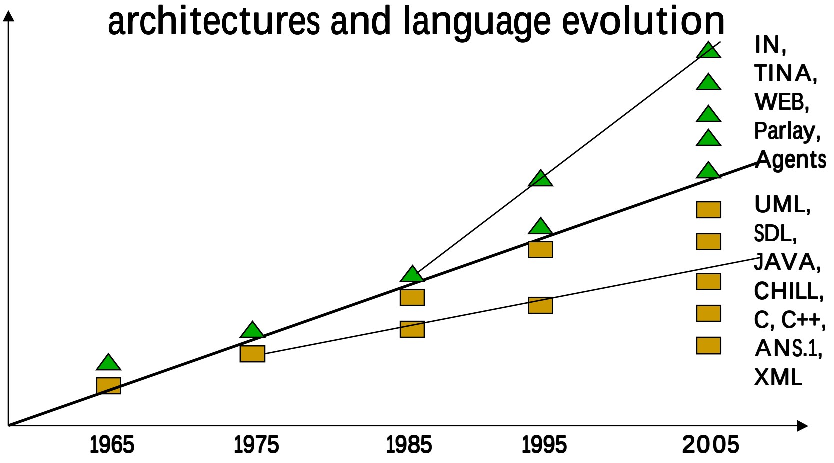
1985

1995

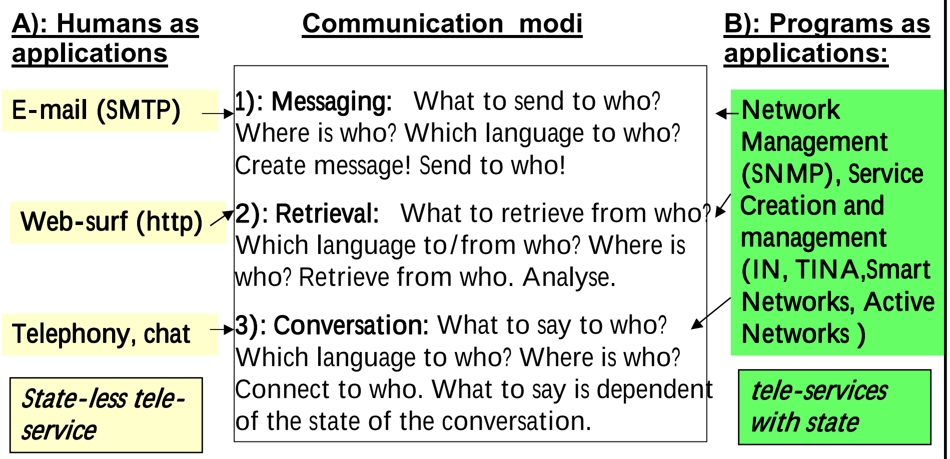
2005

IN: Intelligent Networks (ITUrec.: Q. 1200)
SDL95: ITUs Specification and Description Language
CHILL: ITUs High Level Language

Diversity examples on tele-service architectures and language evolution



Complex tele-services specifications have states



A): Humans as applications

Communication modi

B): Programs as applications:

E-mail (SMTP)

Web-surf (http)

Telephony, chat

State-less tele-service

1): Messaging: What to send to who? Where is who? Which language to who? Create message! Send to who!

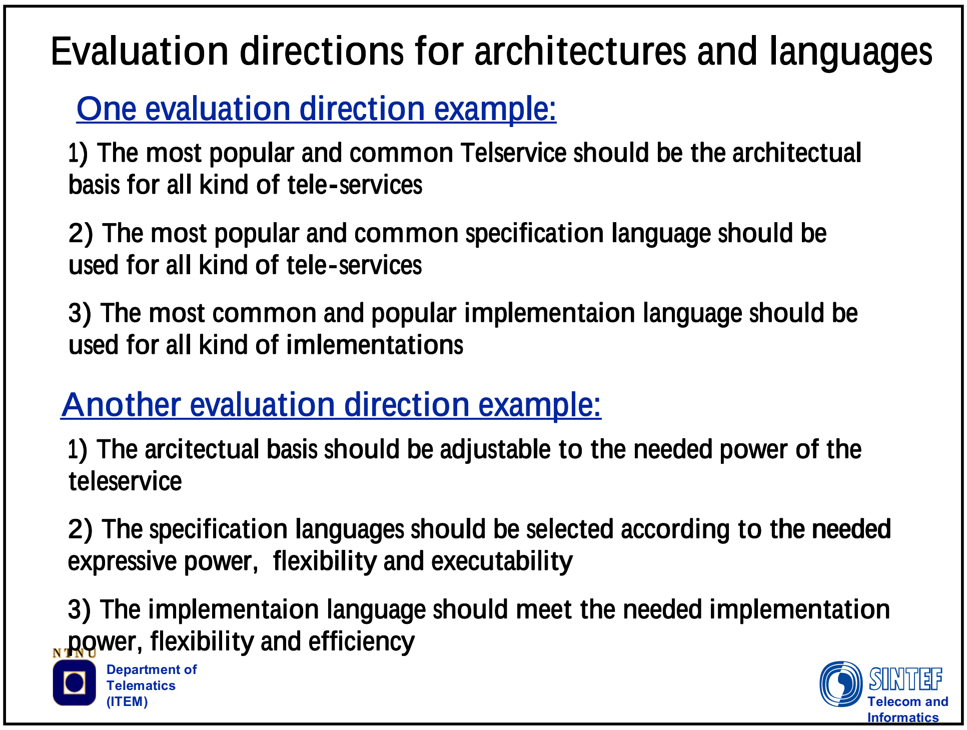
2): Retrieval: What to retrieve from who? Which language to/from who? Where is who? Retrieve from who. Analyse.

3): Conversation: What to say to who? Which language to who? Where is who? Connect to who. What to say is dependent of the state of the conversation.

Network Management (SNMP), Service Creation and management (IN, TINA,Smart Networks, Active Networks)

tele-services with state

The idea about a general state-less functionality cannot be extended beyond its very limited application within the transport service and to tele-services directly supporting Human users.



Evaluation directions for architectures and languages

One evaluation direction example:

1) The most popular and common Telservice should be the architectual basis for all kind of tele-services

2) The most popular and common specification language should be used for all kind of tele-services

3) The most common and popular implementaion language should be used for all kind of imlementations

Another evaluation direction example:

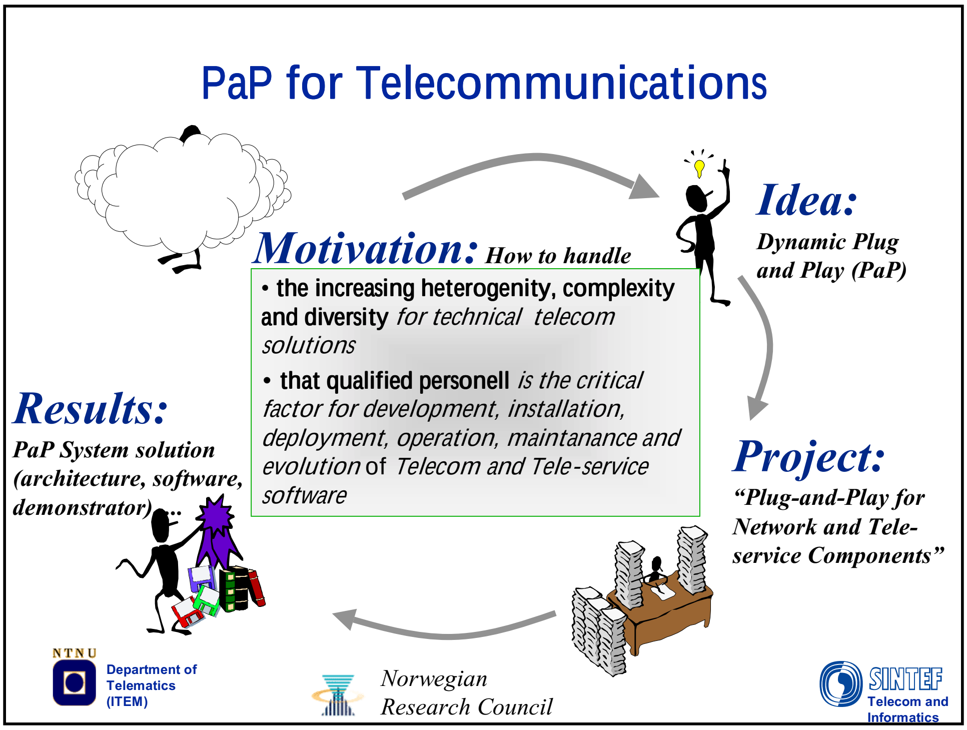
1) The arcitectual basis should be adjustable to the needed power of the teleservice

2) The specification languages should be selected according to the needed expressive power, flexibility and executability

3) The implementaion language should meet the needed implementation power, flexibility and efficiency

NTNU Department of Telematics (ITEM)

SINTEF Telecom and Informatics



PaP for Telecommunications

Motivation: ***How to handle***

- the increasing heterogenity, complexity and diversity *for technical telecom solutions*
- that qualified personell *is the critical factor for development, installation, deployment, operation, maintanance and evolution* of *Telecom and Tele-service software*

Idea:

Dynamic Plug and Play (PaP)

Project:

"Plug-and-Play for Network and Tele-service Components"

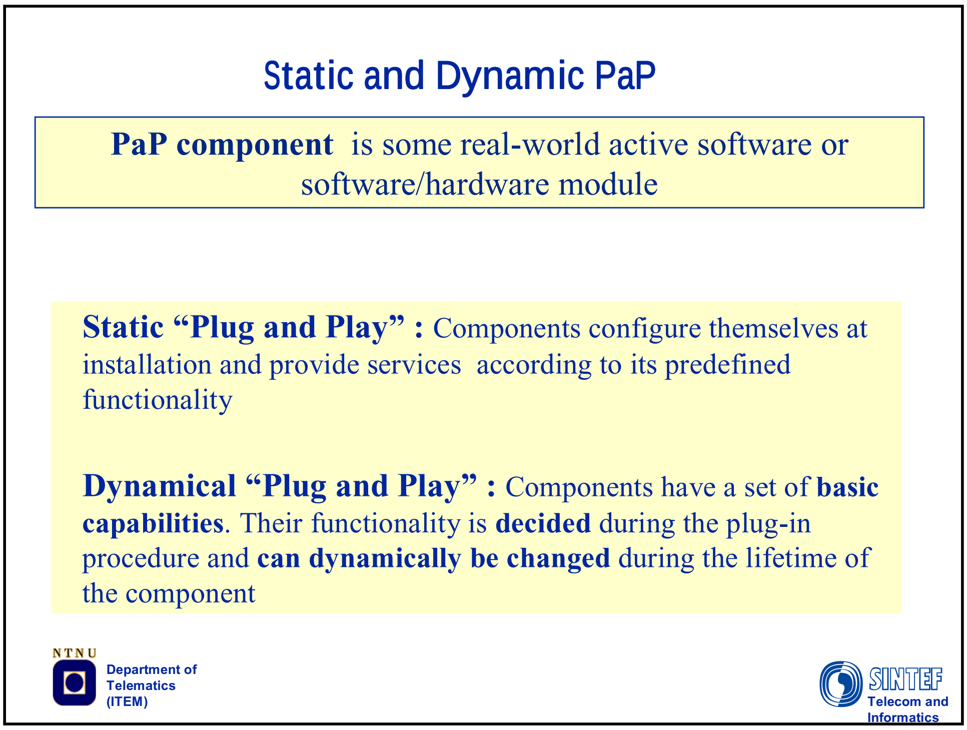
Results:

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

NTNU Department of Telematics (ITEM)

Norwegian Research Council

SINTEF Telecom and Informatics

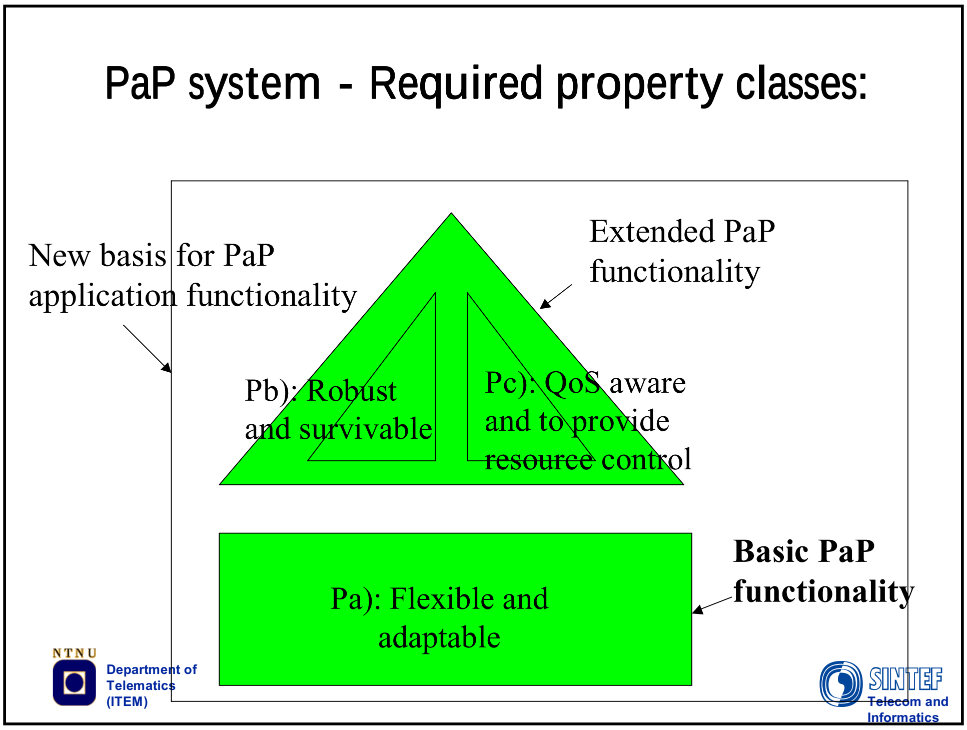


Static and Dynamic PaP

PaP component is some real-world active software or software/hardware module

Static "Plug and Play" : Components configure themselves at installation and provide services according to its predefined functionality

Dynamical "Plug and Play" : Components have a set of **basic capabilities**. Their functionality is **decided** during the plug-in procedure and **can dynamically be changed** during the lifetime of the component



PaP system - Required property classes:

New basis for PaP application functionality
Extended PaP functionality
Pb): Robust and survivable
Pc): QoS aware and to provide resource control
Basic PaP functionality
Pa): Flexible and adaptable

A flexible and adaptable system (Pa) requires:

a system structure and functionality that is not fixed *(adding, moving, removing components and changing component functionality according to needs and capabilities)*

that new components, their external services capabilities and needs are found automatically *(awareness of new components and capabilities, propagation of needed information about changes, propagation of needed new functionality)*

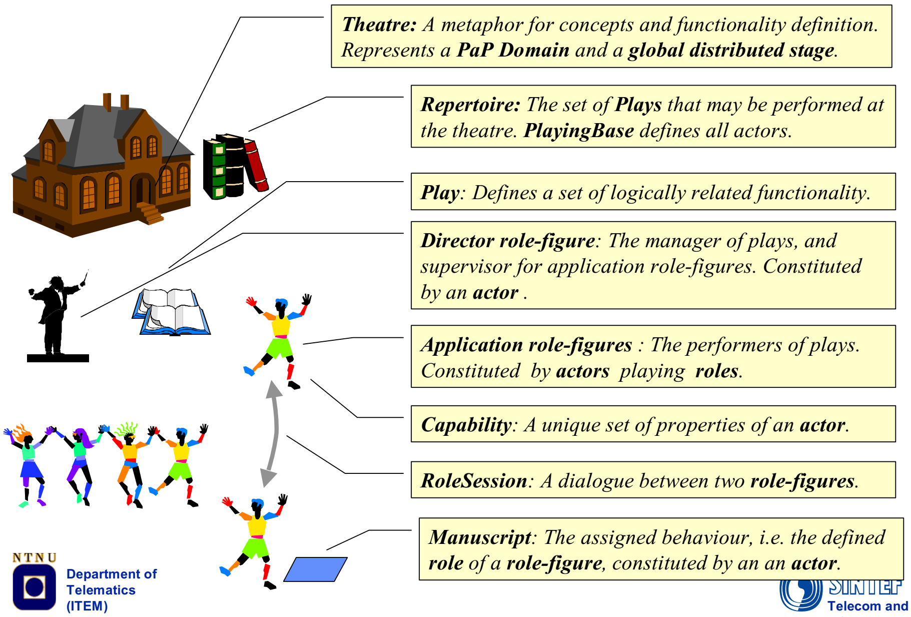
continuous adaption to the environment and operation strategies/policies *(new component functionality, new teleservices, new service and network management functionality, new policy functionality)*

containment and aggregation

NTNU Department of Telematics (ITEM)

SINTEF Telecom and Informatics

The functional architecture is based on a theatre metaphor



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

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 role-figure*: The manager of plays, and supervisor for application role-figures. Constituted by an* ***actor*** *.*

Application role-figures *: The performers of plays. Constituted by* ***actors*** *playing* ***roles****.*

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

RoleSession*: A dialogue between two* ***role-figures****.*

Manuscript*: The assigned behaviour, i.e. the defined* ***role*** *of a* ***role-figure****, constituted by an an* ***actor****.*

NTNU Department of Telematics (ITEM)

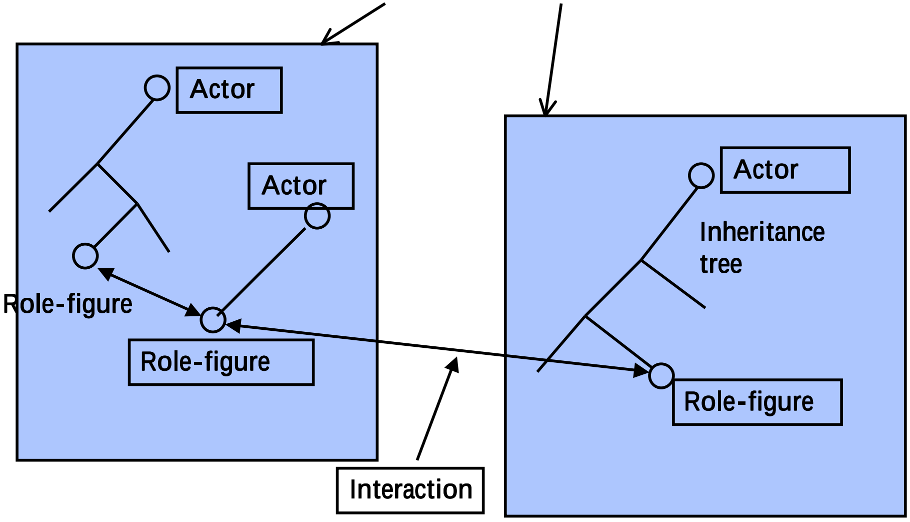
SINTEF Telecom and Informatics

One solution to the basic flexibility and adaptability requirements (Pa)

Actors “implement” **Role-figures** “implement” **PaP-components**

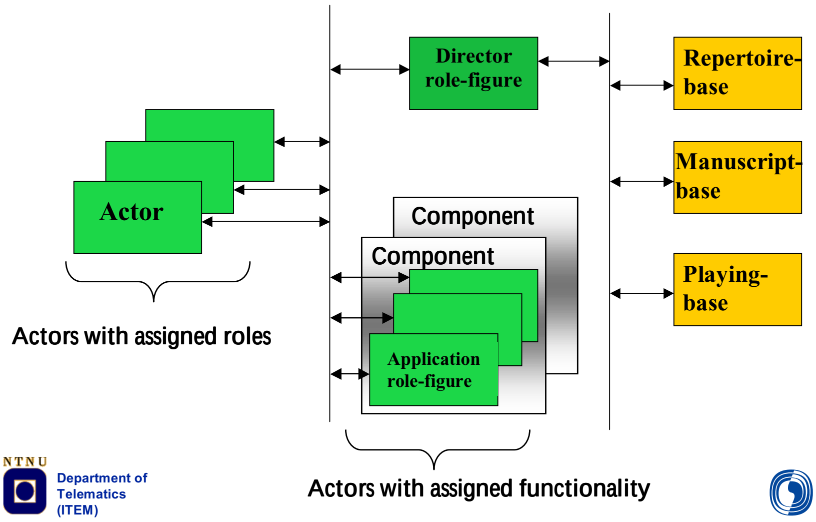
- **PaP-components** are composed from one or more interacting instances of **Role-figures**, which are instances of Role-figure types.
- An **Actor type** is a generic **Role-figure** type. En **Actor** (instance) will execute the functionalty of a Role-figure (type) and then become a **Role-figure** (instance).

PaP-components



One PaP-component can be geographically distributed

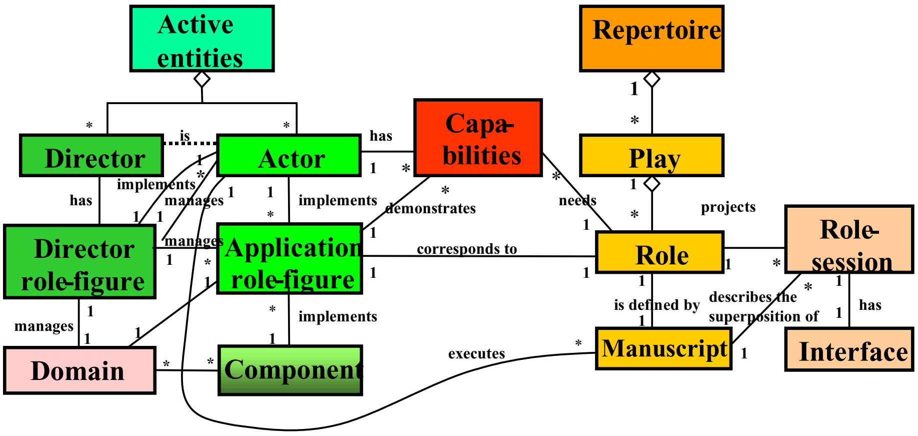
PaP system – Basic instance structure



NTNU Department of Telematics (ITEM)

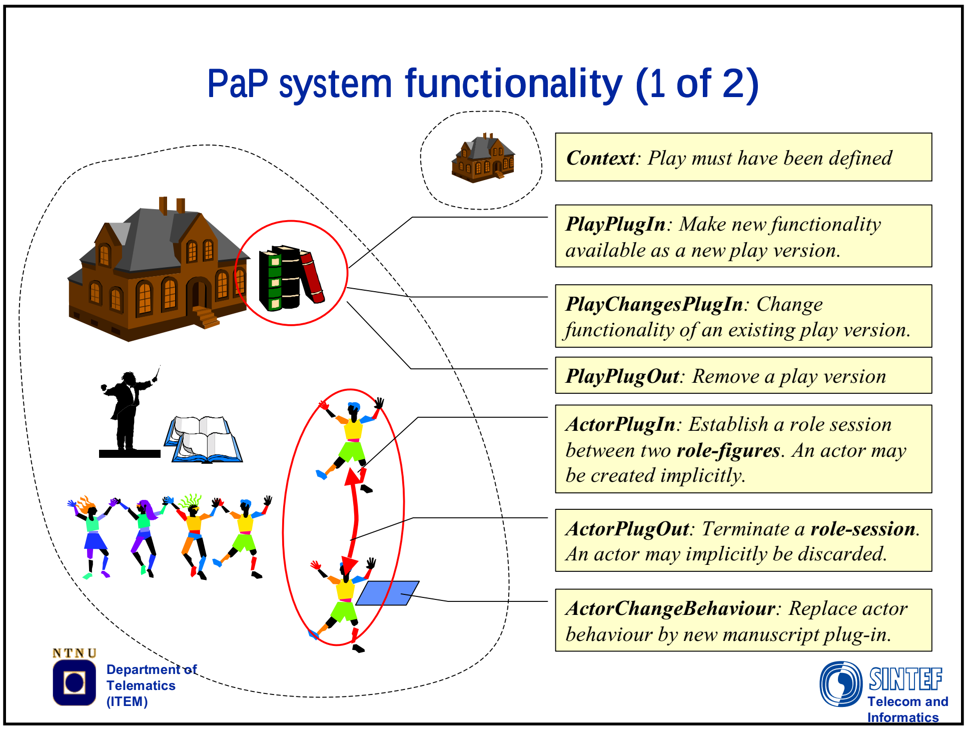
SINTEF Telecom and Informatics

PaP concepts



Legend:

The Director constitutes a Director role-figure, which Role is defined by a Manuscript executed by an Actor



PaP system functionality (1 of 2)

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*

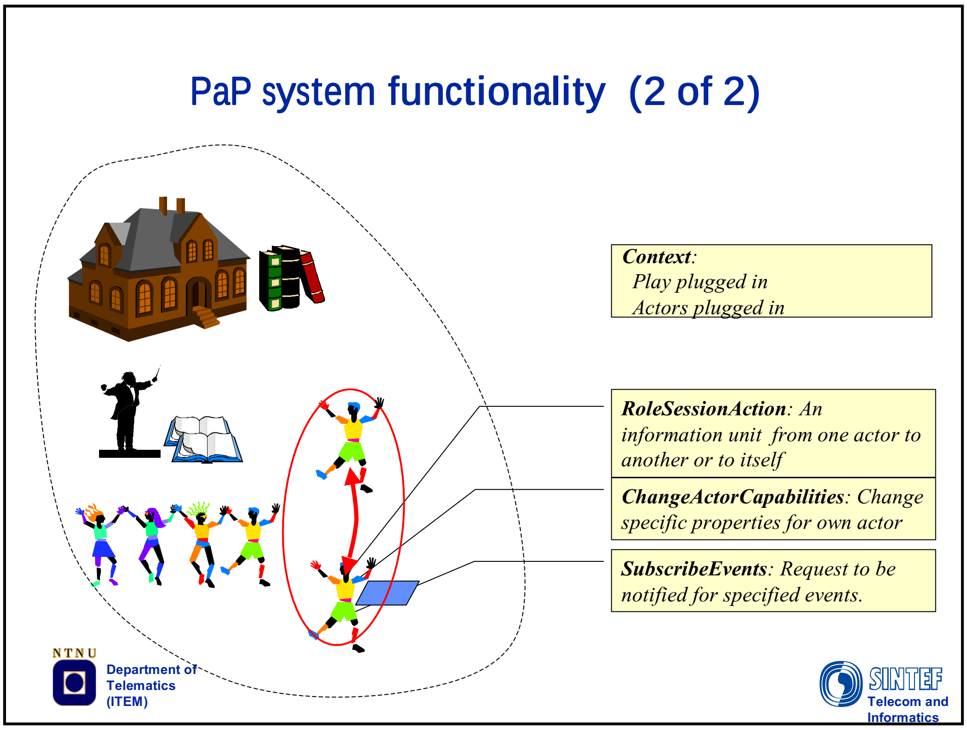
ActorPlugIn*: Establish a role session between two* ***role-figures****. An actor may be created implicitly.*

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

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

NTNU
Department of Telematics (ITEM)

SINTEF
Telecom and Informatics



PaP system functionality (2 of 2)

Context*:*
Play plugged in
Actors plugged in

RoleSessionAction*: An information unit from one actor to another or to itself*

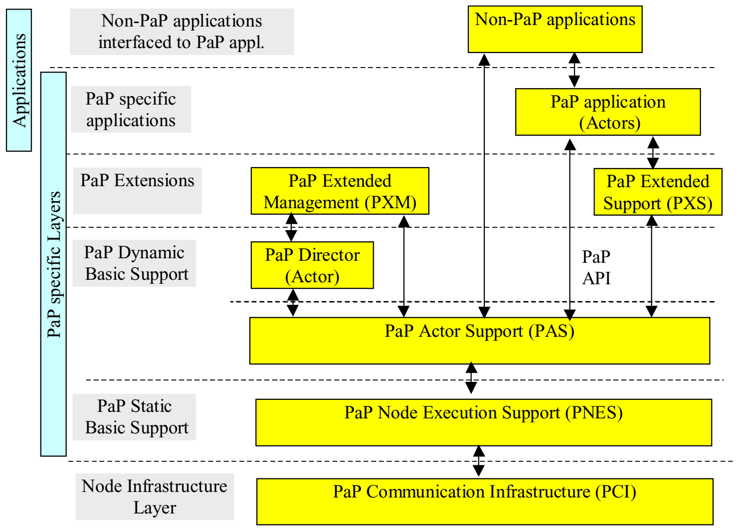
ChangeActorCapabilities*: Change specific properties for own actor*

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

NTNU
Department of Telematics (ITEM)

SINTEF
Telecom and Informatics

The engineering model


Applications

PaP specific Layers

Non-PaP applications interfaced to PaP appl.

PaP specific applications

PaP Extensions

PaP Dynamic Basic Support

PaP Static Basic Support

Node Infrastructure Layer

Non-PaP applications

PaP application (Actors)

PaP Extended Management (PXM)

PaP Extended Support (PXS)

PaP Director (Actor)

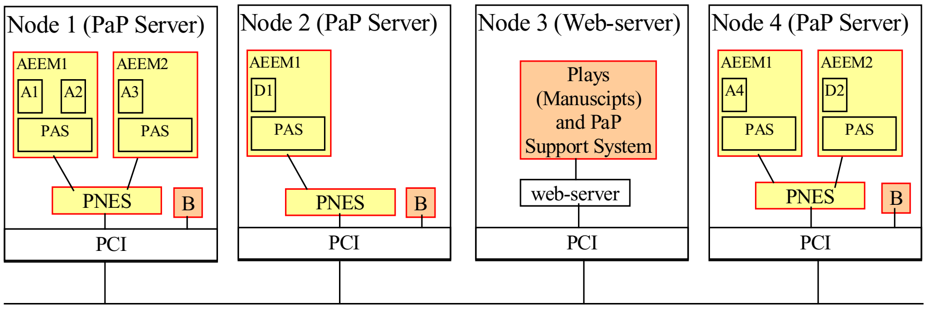
PaP API

PaP Actor Support (PAS)

PaP Node Execution Support (PNES)

PaP Communication Infrastructure (PCI)

A PaP system example


Node 1 (PaP Server)

AEEM1 A1 A2 PAS

AEEM2 A3 PAS

PNES B

PCI

Node 2 (PaP Server)

AEEM1 D1 PAS

PNES B

PCI

Node 3 (Web-server)

Plays (Manuscipts) and PaP Support System

web-server

PCI

Node 4 (PaP Server)

AEEM1 A4 PAS

AEEM2 D2 PAS

PNES B

PCI

Legend:

Ai : Actor no i

Di: Director no i

AEEMi: Actor-environement-execution-module no i

B: PaP Boot

Legend:

🟧 Static available

🟨 Dynamic available

The Java Implementation Model

Java Terms used in PaP implementation

Classes

Used for implementation of all functionality, and also used for grouping of logically related information. Inheritance (‘extent’) is used for specialisation of generalised classes.

Interfaces

Used to gain access to same object instances from different objects. Defines the interfaces between objects and their environments.

Objects

Objects are the instances of classes that defines the executable system.

Threads

Used to separate different activities operating either independent of each other, or activities loosely couplet to each other.

Java RMI

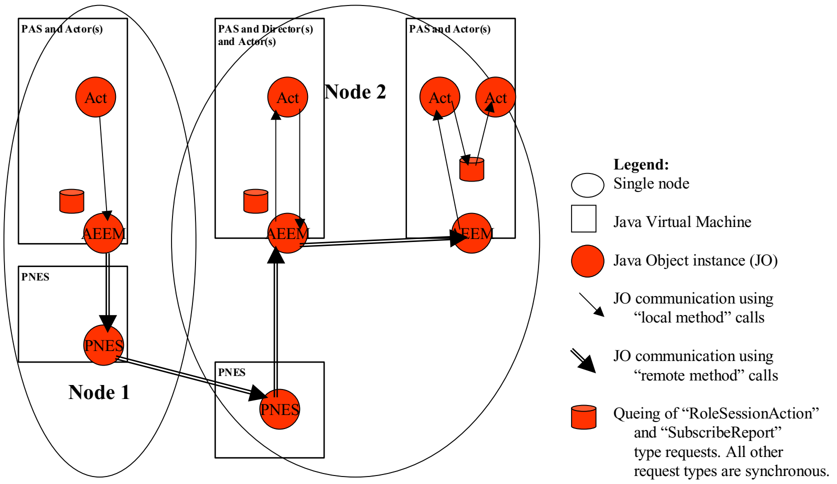
Used as a common basis for communication between objects located within different Java Virtual Machines (JVM). The ‘rmiregistry’ is used for registration and identification of adressable entities of types PNES and AEEM.

NTNU
Department of
Telematics
(ITEM)

SINTEF
Telecom and
Informatics

The Distributed PaP Solution

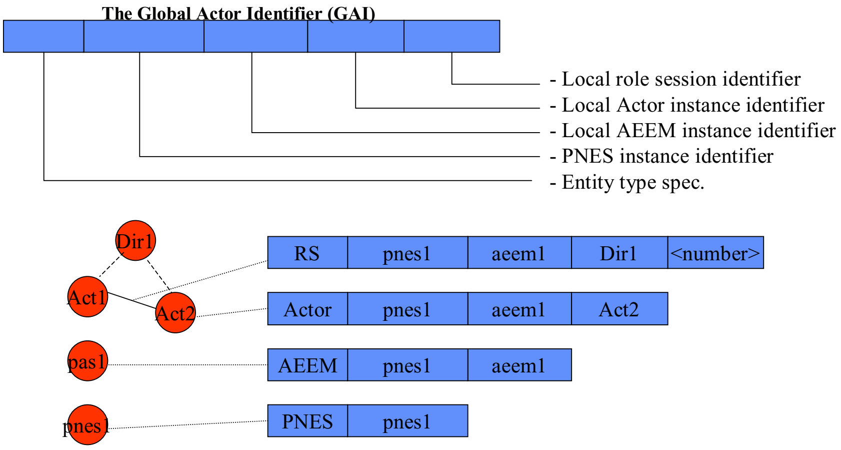
The Synchronous communication model


PAS and Actor(s)
PAS and Director(s) and Actor(s)
PAS and Actor(s)
Node 2
Act
Act
Act
Act
AEEM
AEEM
AEEM
PNES
PNES
PNES
PNES
Node 1
Legend:
Single node
Java Virtual Machine
Java Object instance (JO)
JO communication using “local method” calls
JO communication using “remote method” calls
Queing of “RoleSessionAction” and “SubscribeReport” type requests. All other request types are synchronous.

NTNU
Department of
Telematics
(ITEM)

SINTEF
Telecom and
Informatics

The Distributed PaP Solution
Addressing and routing values

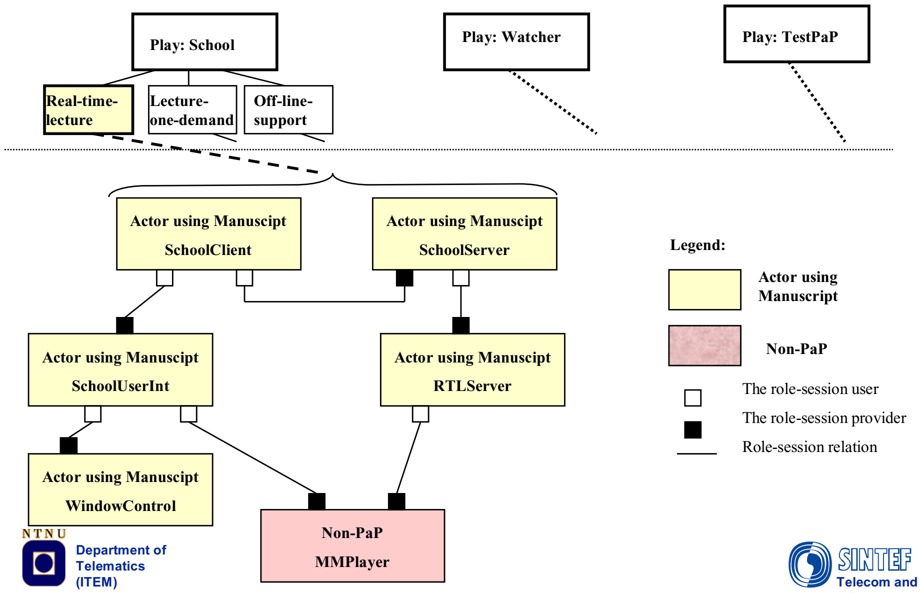

The Global Actor Identifier (GAI)
- Local role session identifier
- Local Actor instance identifier
- Local AEEM instance identifier
- PNES instance identifier
- Entity type spec.
Dir1
Act1
Act2
pas1
pnes1

RS	pnes1	aeem1	Dir1	<number>
Actor	pnes1	aeem1	Act2	
AEEM	pnes1	aeem1		
PNES	pnes1			

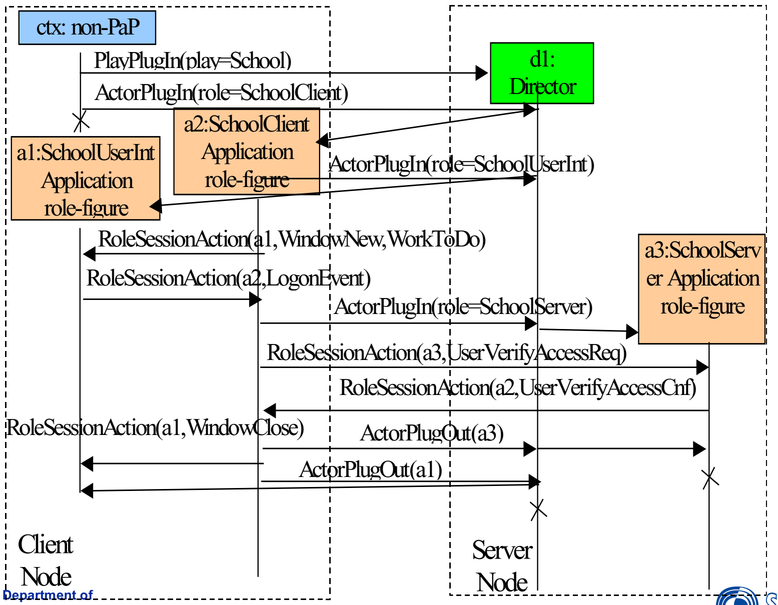
PaP Applications are made for validation and demonstrations

”Tele-School”	-	A Network based learning application
”Watcher”	-	A PaP Support activity monitor
”TestPaP”	-	A tool for automatic testing of PaP Support

Actor structure of Play:Real-time-lecture


Play: School
Real-time-lecture
Lecture-one-demand
Off-line-support
Play: Watcher
Play: TestPaP
Actor using Manuscipt SchoolClient
Actor using Manuscipt SchoolServer
Actor using Manuscipt SchoolUserInt
Actor using Manuscipt RTLServer
Actor using Manuscipt WindowControl
Non-PaP MMPlayer
Legend:
Actor using Manuscript
Non-PaP
□ The role-session user
■ The role-session provider
Role-session relation

Example Interactions between actors


ctx: non-PaP
PlayPlugIn(play=School)
d1: Director
ActorPlugIn(role=SchoolClient)
a2:SchoolClient Application role-figure
a1:SchoolUserInt Application role-figure
ActorPlugIn(role=SchoolUserInt)
RoleSessionAction(a1,WindowNew,WorkToDo)
RoleSessionAction(a2,LogonEvent)
a3:SchoolServer Application role-figure
ActorPlugIn(role=SchoolServer)
RoleSessionAction(a3,UserVerifyAccessReq)
RoleSessionAction(a2,UserVerifyAccessCnf)
RoleSessionAction(a1,WindowClose)
ActorPlugOut(a3)
ActorPlugOut(a1)
Client Node
Server Node

Summary and conclusions

The PaP architecture has potential to improve software development, deployment, installation, operation, maintenance and evolution for for complex telecommunication and tele-service functionaloty

The specified flexibility and adaptability requirements has been met and demonstrated

PaP solutions is based on available and portable technology (Java). Light weight solution for distributed, asynchronous, message based "soft" real-time applications. Executable software and documentation available at Web: *http://www.item.ntnu.no/~plugandplay*

Ongoing research related to extentions of the PaP architcture to meet the requirement classes Pb) and Pc). Present Dr.ing research topics:
1): Teleservice modelling, 2): Fault tolerance and intrusion prevention,
3): Mobility and 4):Capability handling.

Advantages of using PaP (1 of 2)

1): Development of PaP Applications

Flexibility in application modelling
("Composition" of Plays and Manuscripts from Role-sessions)

Transparency in distributed solutions *(Use of Java/RMI)*

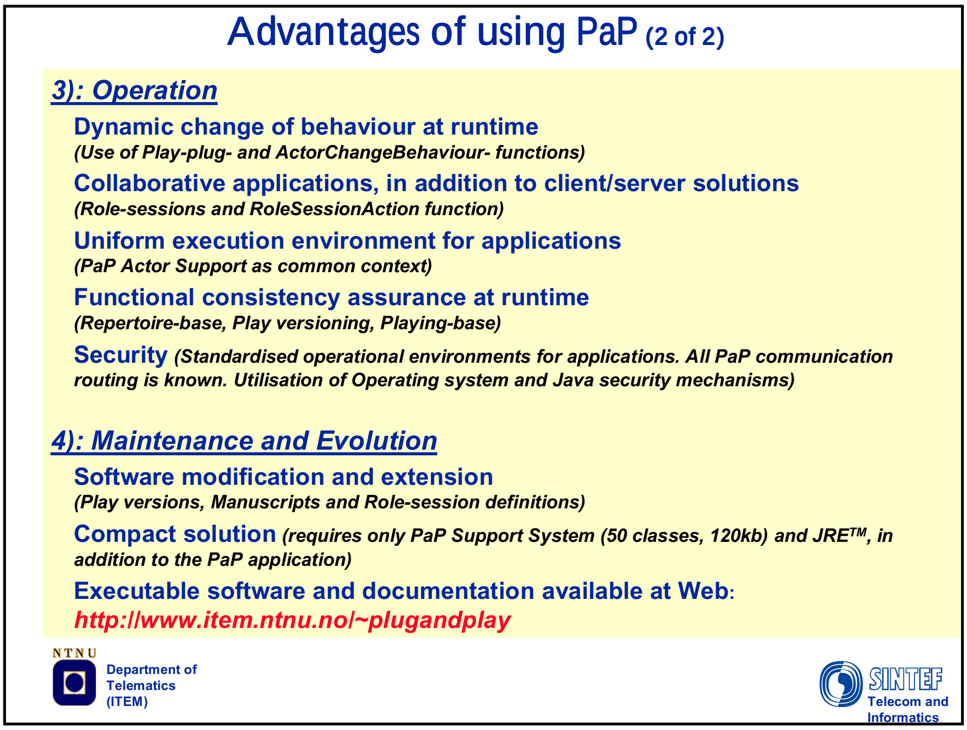
Portable *(Use of Java)*

Mobile agents become possible *(Uniform operational context for Actors, Java)*

Easy monitoring and controlling
(Almost all PaP function requests served by Director)

2): Deployment and Installation

Easy installation and maintenance of installations
(Web-server, PlayPlugIn, PlayChangesPlugIn and PlayPlugOut functions)



Advantages of using PaP (2 of 2)

3): Operation

Dynamic change of behaviour at runtime
(Use of Play-plug- and ActorChangeBehaviour- functions)

Collaborative applications, in addition to client/server solutions
(Role-sessions and RoleSessionAction function)

Uniform execution environment for applications
(PaP Actor Support as common context)

Functional consistency assurance at runtime
(Repertoire-base, Play versioning, Playing-base)

Security *(Standardised operational environments for applications. All PaP communication routing is known. Utilisation of Operating system and Java security mechanisms)*

4): Maintenance and Evolution

Software modification and extension
(Play versions, Manuscripts and Role-session 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

NTNU
Department of
Telematics
(ITEM)

SINTEF
Telecom and
Informatics