Support Specification and Selection in TAPAS

Finn Arve Aagesen, Chutiporn Anutariya, Mazen Malek Shiaa and Bjarne E. Helvik
Department of Telematics
Norwegian University of Science and Technology (NTNU)
N-7491 Trondheim, Norway
{Finn.Arve.Aagesen, Chutiporn.Anutariya, Mazen.Malek.Shiaa, Bjarne.E.Helvik}@item.ntnu.no

Abstract

A theoretical framework for support specification and selection in Plug-and-Play (PaP) architecture is proposed with a representation, computation and reasoning mechanism for semantic description and matching of support required by a particular PaP service system and support offered by a running PaP system. By analysing such given required and offered support, the framework allows appropriate service system configurations, satisfying all the specified constraints and requirements, to be automatically generated. Moreover, its integration with optimisation rules, resource consumption specifications and QoS measurement techniques can also lead to generation of optimal (re)configurations, suggesting which node in the system should constitute which service component, in order to achieve mandated performance levels and at the same time be able to meet user satisfaction.

1 Introduction

TAPAS (*Telecommunication Architecture for Plug and Play Systems*) [1] is based on generic actors in the nodes of the network that can download manuscripts defining roles to be played. However, the ability to play roles depends on the *defined required capability* and the *matching offered capability* in a node where an actor is going to play. Examples of capabilities are *processing and communication resources* such as CPU and transmission channels, *standard equipment* such as printers and media handling devices, *special equipment* such as encrypting devices, and *data* such as user login and access rights.

Besides matching of offered and required capabilities, the status of the running system and the status required by a role have to be taken into consideration

A play defining functionality of a particular PaP service system consists of several actors playing different roles, each probably having requirements on both available capabilities and status, which are here denoted as required support. Hence, such questions as "where to run the play and what is the optimal configuration for the play?" may arise. This paper focuses on development of a solid XML-based framework for support specification and selection in TAPAS with a well-established infrastructure for reasoning, configuration and reconfiguration of the system by employment of *XML Declarative Description* (*XDD*) modelling language [4].

Sect. 2 introduces PaP systems and defines the concepts of capabilities and status, Sect. 3 discusses significant functions in support management, Sect. 4 proposes a framework for support specification and selection—a support management function, Sect. 5 demonstrates the mechanisms of the developed approach by means of TeleSchool application example, and Sect. 6 concludes and presents further research direction. Appendix recalls fundamental definitions and concepts of the XDD theory.

2 Capabilities and Status

As illustrated by Figure 1, a *PaP system* consists of PaP *service systems***,** which are units related to some well-defined functionality. A PaP service system can be decomposed into *service components*. A service component is realised by a *role figure* based on a *role* defined by a *manuscript* and is executed by an *actor*.

A role figure, however, is realised in an executing environment in a node and is utilising capabilities. A capability is an inherent property of a *capability component.* These concepts of service components and capability components reflect *two different viewpoints* of the PaP system. A capability component may have several capabilities. These capabilities are offered to actors which constitute role-figures in various plays.

Basically, capabilities can be classified into:

- *Functions*: pure software or combined software/ hardware components used for performing particular tasks,
- *Resources*: hardware components with finite capacity, such as processing, storage and communication units,

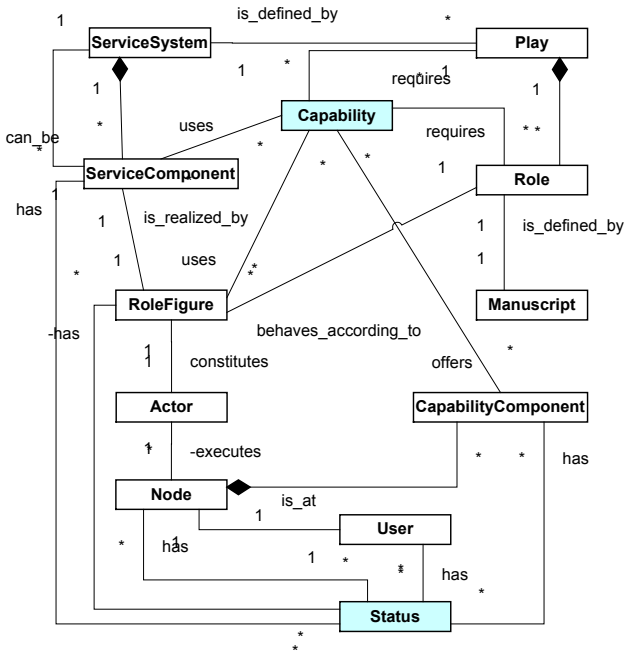

Figure 1. Basic PaP conceptual model.

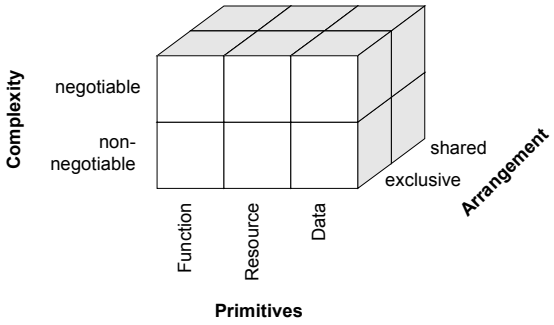

Figure 2. Classification attributes of capabilities.

- *Data*: just data, the interpretation, validity and life span of which depend on the context.

Note that the power of each capability function and resource is measured by its respective QoS characteristics.

Besides these three primitives, capabilities can also be classified by their complexities, i.e., whether they are negotiable or not. For example, transmission function is a negotiable capability, while access right data is not. Moreover, capabilities can, by arrangement or use, be exclusive or shared. For instance, transmission channels and Web servers are by nature shared resources, a password is by nature exclusive information, and access rights are exclusive or shared.

Figure 2 presents such a three-dimensional view of capabilities.

While both capabilities and plays are specifications of what can be done, *status* is, at a certain time instant, the situation in a PaP system with respect to the actual number of nodes, playing plays, traffic situation, etc. Status can both comprise observable counting measures, measures for QoS or calculated predicates related to these counts and calculated measures. It reflects the resulting state of the sytem, which cannot directly be changed and negotiated.

Capabilities and status are here denoted as *support*. Roles can both have requirements on available capabilities and status (i.e., requirements on support). A support requirement of a play is therefore the resulting requirements of the roles constituting it.

3 Support Management

Figure 3 gives an overview picture of support management functionality, which comprises the following main functions:

- *Capability Installation*: for *installation* and *de-installation* of capability components,
- *Capability Handling*: for updating a view of offered capabilities and for dynamic capability allocation,
- *Status Monitoring*: for provision of a view of offered status,
- *Support Selection*: for determination and optimisation of a PaP service system configuration by analysing support offered by the executing PaP system, support required by a play to be installed as well as QoS of the resulting system, and
- *Service Installation*: for deployment and invocation of a service system.

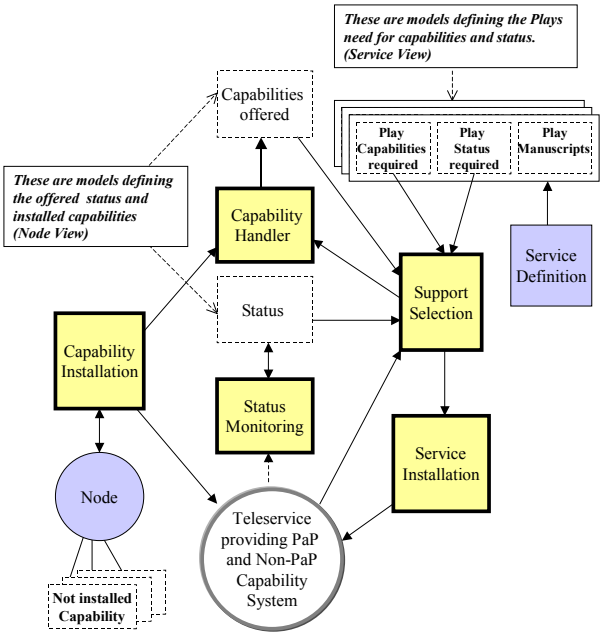

Figure 3. Support management functionality.

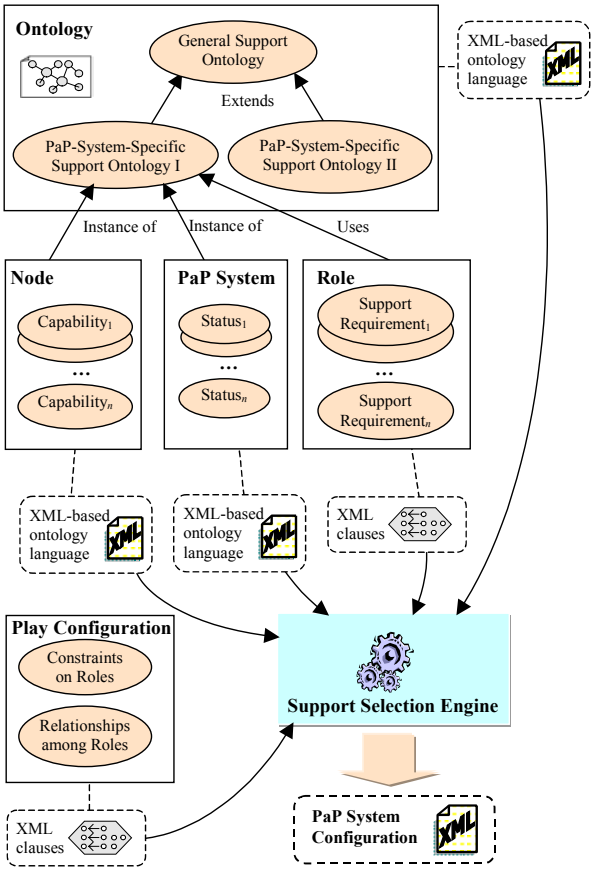

Figure 4. A framework for support specification and selection.

With emphasis on the support selection function, a framework for modelling support offered by a system and support required by a play and for generating appropriate system configurations will be outlined next.

4 Support Specification and Selection

4.1 Overview of the Framework

Figure 4 depicts a framework for support specification and selection in TAPAS. It defines *General Support Ontology* as a top-level conceptual model for semantic description of capabilities and status of PaP systems, enabling each application-specific PaP system to extend it by introducing and defining new domain-specific capabilities, status as well as their properties and relationships. Therefore, a capability of each node in a PaP system together with the system's status can be immediately described as instances of the ontology. Both ontology definitions and instances can be represented by any XML-based ontology modelling language, such as *UML class and object diagrams* or *DARPA Agent Markup Language + Ontology Inference Layer* (*DAML+OIL*) [2]. However, these ontology languages merely provide a set of predefined modelling constructs, such as subclassOf, minCardinality, maxCardinality, while lacking an ability to represent inherent interrelationships and complex constraints on elements in a domain. Thus, their mechanisms are insufficient for describing support requirement and play configuration. *XML Declarative Description* (*XDD*) theory [4]—an expressive XML-based knowledge representation language with well-defined semantics and reasoning mechanisms (cf. Appendix for a review of the theory)—is employed to overcome this limitation.

Given a play comprising various roles, *XML clauses* can be employed not only to represent support requirements of each individual role, but also to describe the play's compositional constraints as well as relationships among those roles. Selection of nodes playing particular roles is also materialized by appropriate formalization of XML clauses.

In the next subsections, mechanisms for modelling each component in the framework and for computing a list of possible PaP service system configurations will be elaborated.

4.2 Support Ontology

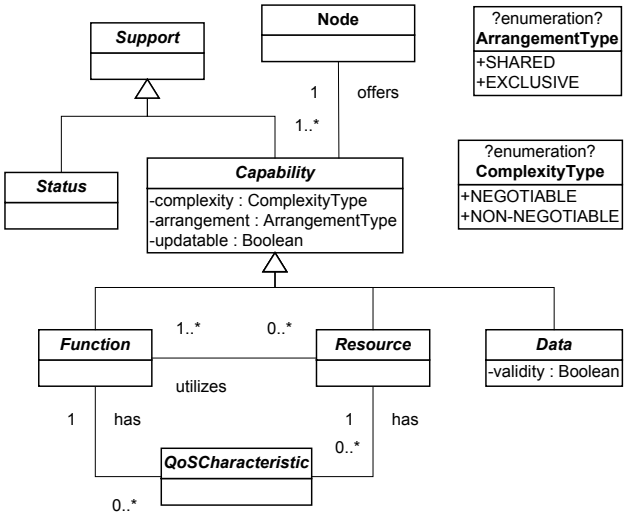

Figure 5. General support ontology.

By means of UML class diagrams, Figure 5 presents the defined general support ontology. However, due to space limitation, its corresponding representation in *XML Metadata Interchange* (*XMI*) [3] format will not be given. The ontology specifies that Status and Capability are subclasses of Support, and a node in a system may offer one or more capabilities. Capability has three important attributes: complexity, arrangement and updatable, used for characterising whether a capability is negotiable or non-negotiable, shared or exclusive, and updatable or not, respectively. Moreover, as discussed previously, Capability can be spe-

cialised into the three subclasses: Function, Resource and Data. Function may utilize zero or more Resources, and both Function and Resource may have certain set of QoSCharacteristics. Data, on the other hand, has a boolean attribute: validity for specifying whether the data is valid or not.

Each specific PaP system can extend this general support ontology for definition of its own conceptual support ontology. An example of such ontology will be seen in next section. Moreover, it should be noted that instead of defining a new ontology for every PaP service system, a standard, predefined one, if available, such as those various IETF MIBs, can also be shared and reused.

4.3 Capability Specification

Specifications of capabilities offered by nodes in a system are simply modelled as instances of the system's support ontology. If UML class diagram is employed to model a support ontology, these capability specifications are readily described by *object diagrams* comprising sets of related instances or objects of respective capability classes with appropriate initialised attribute values.

Table 1. PaP support specification and selection.

Modelling Components	Modelled by
1. Support Ontology	XML-based ontology language
2. Capability Specification	XML-based ontology language
3. Status Specification	XML-based ontology language
4. Support Requirement Specification	XML clauses
5. Play Configuration Specification	XML clauses

** **Support Specification and Selection** is modelled as an XDD description, comprising XML documents and XML clauses representing these components. The declarative semantics of the description yields possible system configurations.

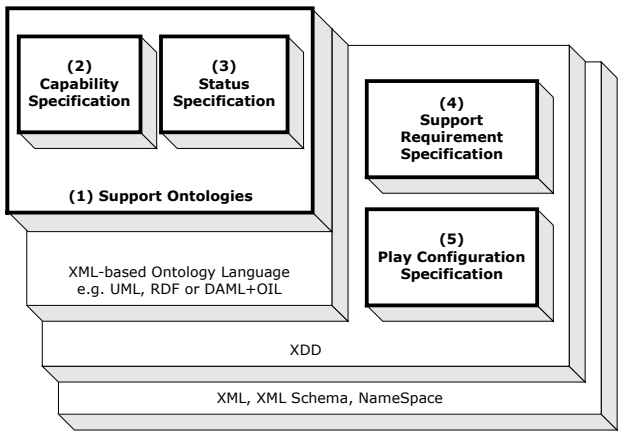

Figure 6. Representation language layer.

4.4 Status Specification

Similar to capability specification, status of a system at a certain time can also be described by instances of the system's support ontology.

4.5 Support Requirement Specification

Support requirement specification of a certain role in a play is expressed by appropriate XML clauses, the head of which specify the role to be played and the body of which describe the required system status and the capabilities of a node for fulfilling such a role.

4.6 Play Configuration Specification

A play configuration specification is represented as a corresponding set of XML clauses, the head of which identify components of the play, while the body of which describe the configuration restrictions.

As summarized by Table 1 and Figure 6, support specification and selection in a PaP system is modelled as an XDD description, the meaning of which yields a list of possible configurations of the system, stating which node could play which role.

5 Example: TeleSchool Application

This section demonstrates the proposed mechanisms for support specification and selection by means of TeleSchool application example. Assume that a TeleSchool play comprises a set of manuscripts describing the behaviours and actions of the three roles:

- *TeleSchool Server*: providing real-time lecture and archived lecture services,
- *Real-time Client*: allowing students to participate available real-time lectures,
- *Archived Lecture Client*: providing facilities for students to go through archived lectures.

Moreover, assume that a TeleSchool play configuration must satisfy the constraints:

- A node in the application can execute zero ore more actors, each possibly constituting different role figures;
- There exist exactly a dedicated TeleSchool server and zero or more clients;

5.1 Capability Ontology

Based on the defined general support ontology of Figure 5, Figure 7 models a system-specific capability ontology, specifying TeleSchool-related concept hierarchies of the three capability types: resources, functions and data, and also identifies their inherent properties and QoS characteristics. Note that the given ontology is only for a demonstration purpose and does not present a com-

plete model for TeleSchool system, i.e., such capabilities as transmission channels, CPU, encryption function, for instance, are omitted.

In this ontology example, there are two subclasses of Function: TransmissionFunction and ProcessingFunction. The former has two QoSCharacteristics: Delay and Bandwidth, while the latter is specialised into MediaProcessing, which is further specialised into VDOCapture and VDOPlayer. VDOCapture is defined as a negotiable and exclusive capability with three QoSCharacteristics: CaptureRate (how many frames per second), CaptureResolution, and AudioCaptureQuality (what are the offered bitQuantisation and kHzSamplingRate).

Focusing on Resource, the ontology defines PhysicalStorage as a type of Resource, having DiskCapacity QoS characteristic. Harddisk is here defined as a subclass of PhysicalStorage; hence it will also derive DiskCapacity QoS characteristic from its superclass.

IPAddress and UserRelatedData are defined as Capability Data. User and UserGroup are UserRelatedData with a binary relation, specifying that a User may be a member of one or more UserGroups.

5.2 Capability Specification

Figure 8 gives a specification of capabilities offered by the three nodes n1, n2 and n3 in the system.

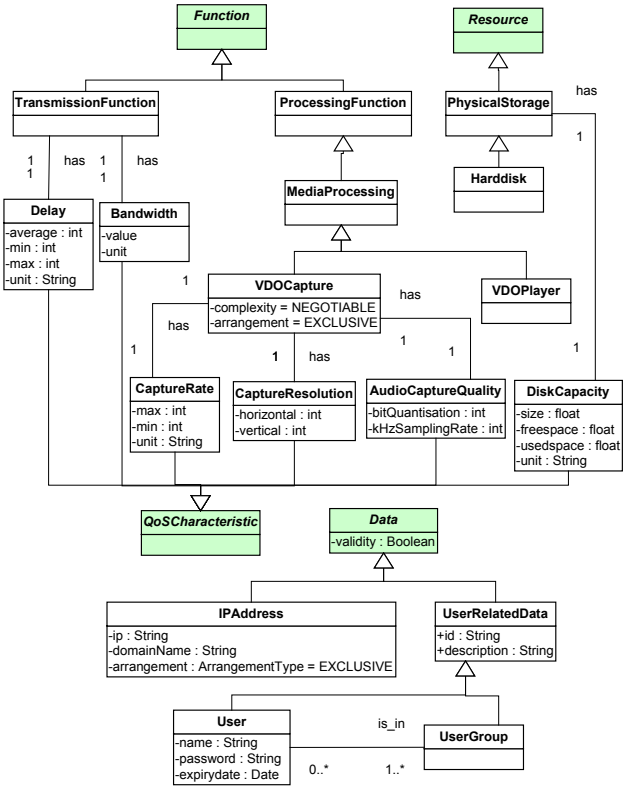

Figure 7. UML-based TeleSchool capability ontology.

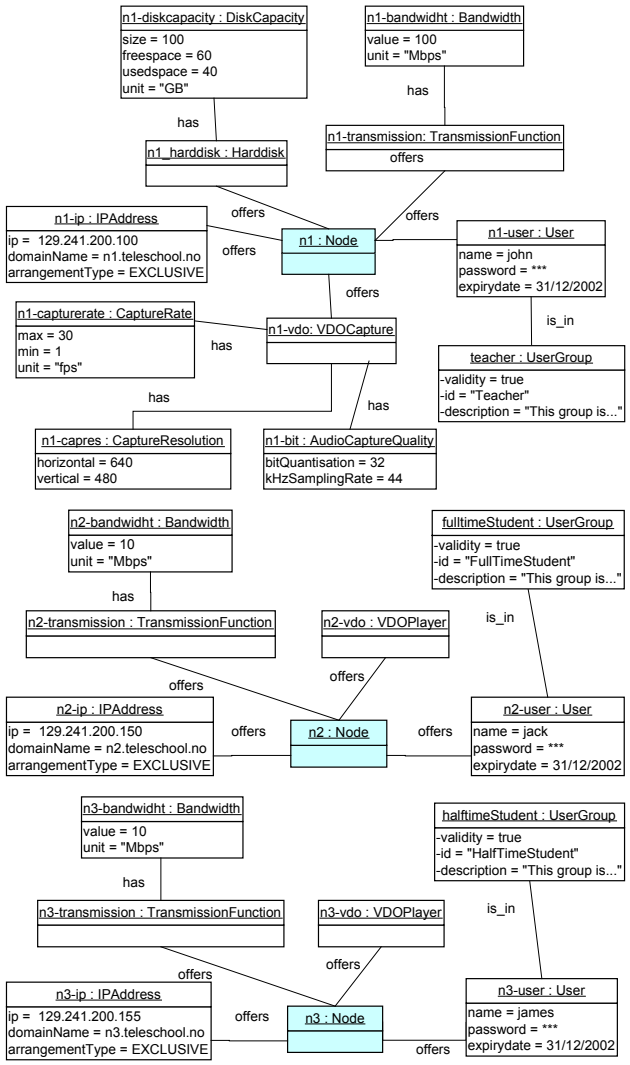

Figure 8. Offered capability specification.

5.3 Capability Requirement Specification

XML clauses formulating capability requirements of the three roles in TeleSchool system will be given. However, for ease of understanding, these clauses will be presented graphically using UML class diagram notations instead of encoding in UML-XMI format. Recall that variables in XML clauses are preceded with '$', followed by their types and their names. For example, **$S**:usergroup denotes a **S**tring-variable instantiable into a string, while **$E**:userAttrs an **E**xpression-variable instantiable into a list of UML classes, objects or attributes. When it is clear from the context, variable types may be omitted. Moreover, those variables beginning with '$?' represent anonymous variables. Recall also that the head of a clause intuitively models the consequence part, while the body describes the antecedence or the condition part.

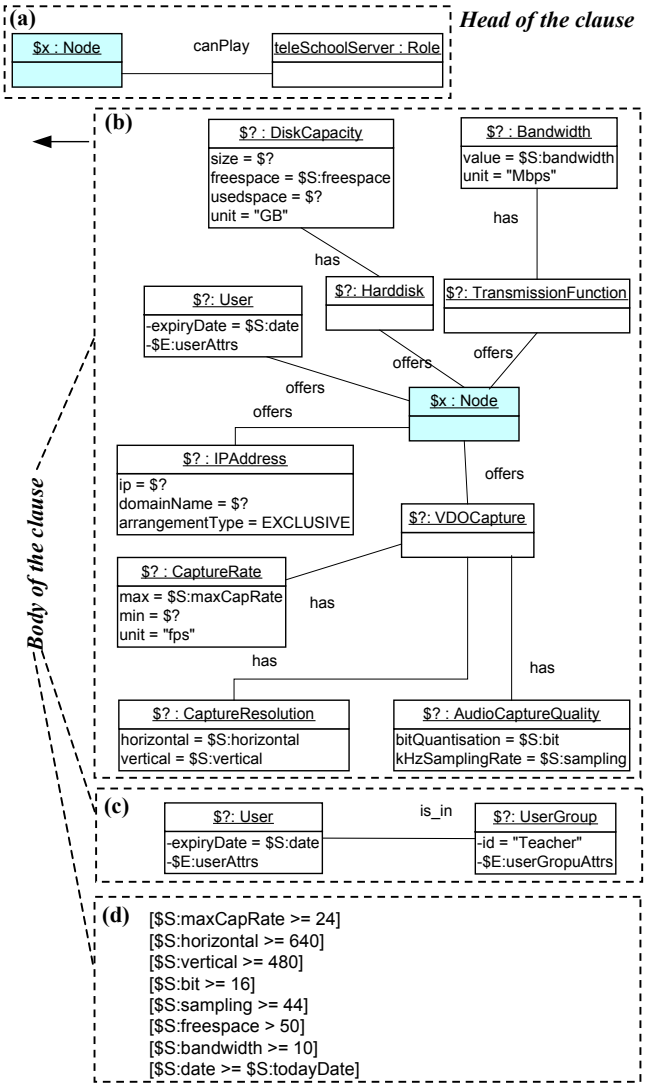

Figure 9. Requirement for TeleSchoolServer Role.

The clause of Figure 9 specifies a capability requirement for the teleSchoolServer role, which can be read as follows:

(a) Any instance $x of the class Node can play the teleSchoolServer role,

if

(b) $x offers the following capabilities

- a logging-on User,
- an IPAddress,
- a VDOCapture function,
- a Transmission function,
- a Harddisk,

(c) the logging-on User is a member of the TeacherGroup, and

(d) the following conditions on properties and QoS characteristics of certain capabilities are satisfied:

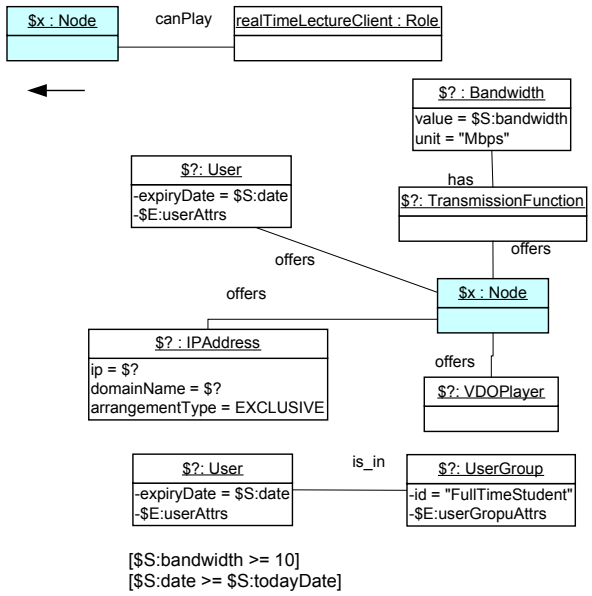

Figure 10. Requirement for RealTimeClient Role.

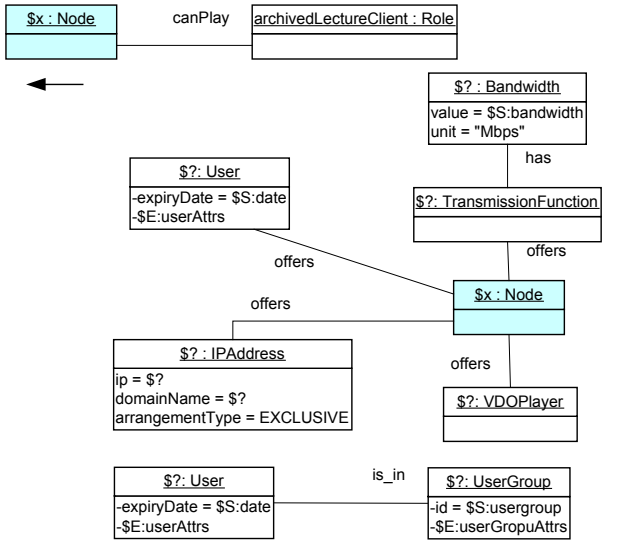

Figure 11. Requirement for ArchivedLectureClient Role.

- [$S:maxCapRate >= 24] : the VDOCapture function has at least 24 frame/sec. capture rate,
- [$S:horizontal >= 640] and [$S:vertical >= 480] : the horizontal and vertical dimension (resolution) of the VDOCapture function are at least 640 and 480, respectively,
- [$S:bit >= 16] and [$S:sampling >= 44] : the captured audio quality is at least at 16-bit quantisation and 44 kHz sampling-rate,

- [$S:bandwidth >= 10] : the data transmission capacity is greater than or equal to 10 Mbps,
- [$S:freespace > 50] : the Harddisk has more than 50 GB free disk-space,
- [$S:date >= $S:todayDate] : the user account has not yet been expired.

The clauses given by Figures 10 and 11, on the other hand, model requirements for the realTimeClient and archivedLectureClient roles, respectively, by restricting that any node $x can constitute such a role, if it has an IPAddress, a VDOPlayer function, a Transmission function with at least 10 Mbps, and an unexpired User account. Moreover, to play the realTimeClient role, the logging-on User account must be a member of FullTimeStudent group, while the archivedLectureClient demands that the User must be a FullTimeStudent or HalfTimeStudent.

5.4 Play Configuration Specification

Figure 12 formulates TeleSchool play constraints. The head of the clause C_1 defines that a play consists of three roles: teleSchoolServer, realTimeClient and archivedLectureClient, where there exist

- exactly one node, represented by $x, constituting the teleSchoolServer role,
- zero or more nodes, represented by $E:realTimeClientNodes, playing the realTimeClient role, and
- zero or more nodes, represented by $E:archivedLectureClientNodes, playing the archivedLectureClient role.

The body of C_1 constrains that such node $x must be able to play the teleSchoolServer role (i.e., $x must satisfy all of its capability requirements, defined by the clause of Figure 9).

The constraints notMember($x, $E:realTimeClientNodes) and notMember($x, $E:archivvedLectureClientNodes) ensure that $x will not play other roles, since it must be dedicated to the teleSchoolServer role.

The configuration of the set of nodes represented by $E:realTimeClientNodes and $E:archivedLectureClientNodes will be further restricted by the clauses $C_2 - C_6$. These clauses specify that each node in $E:realTimeClientNodes and $E:archivedLectureClientNodes must satisfy the requirements of the realTimeClient and archivedLectureClient roles, defined by the clauses of Figures 10 and 11, respectively.

5.5 Play Configuration Result

Let an XDD description D comprise XML documents and XML clauses corresponding to those specifications given by Figures 5, 7–12. According to the definition of the declarative semantics of XDD descriptions, the meaning of the description D yields a list of possible configurations for the TeleSchool play. Figure 13 presents three of them. Configuration 1, for example, specifies that the node n1 plays the teleSchoolServer role, n2 realTimeClient, and n3 archivedLectureClient.

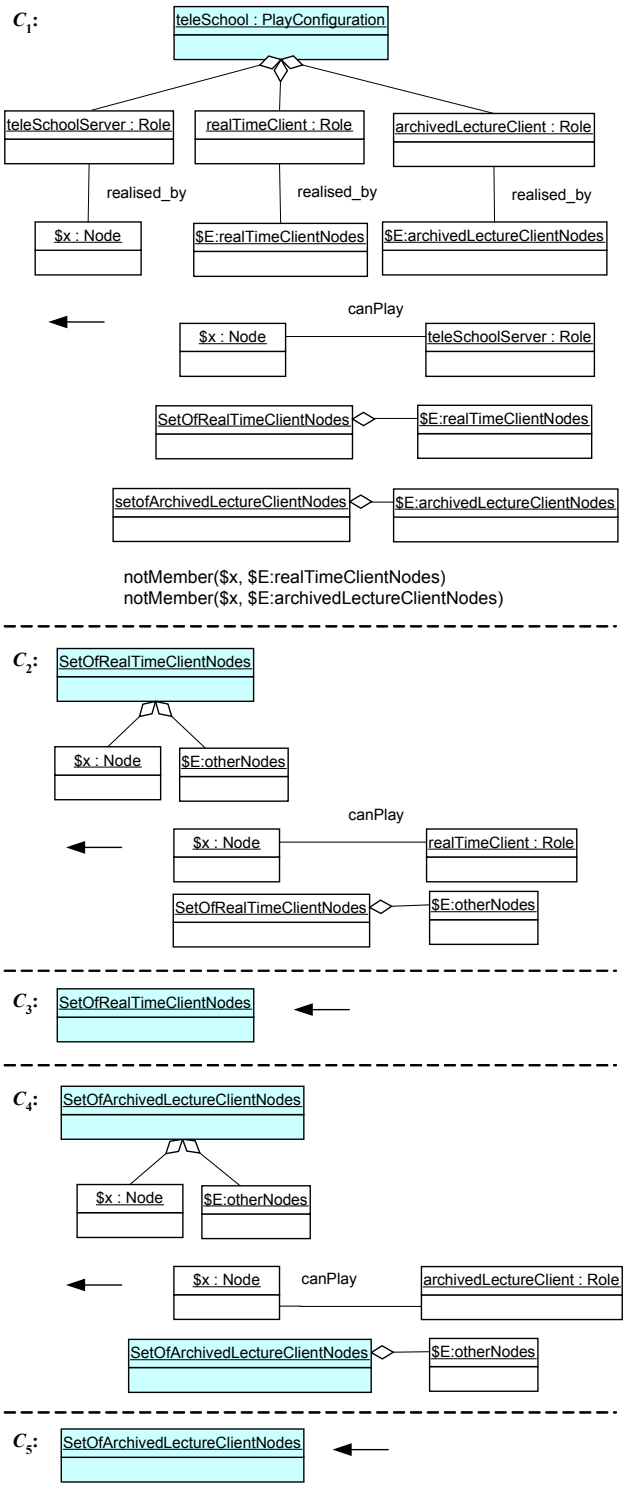

Figure 12. Play configuration specification.

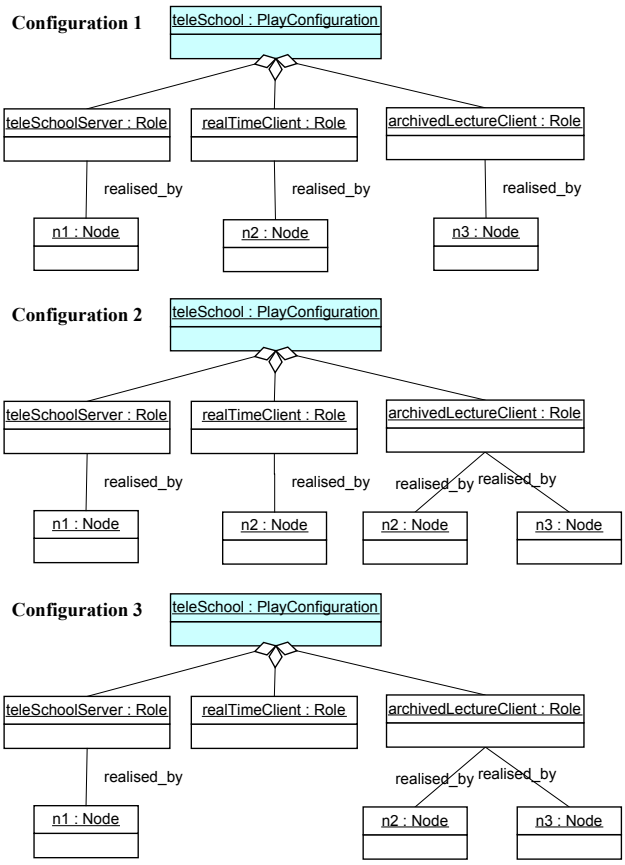

Figure 13. Examples of obtained configurations for TeleSchool play.

6 Conclusions

The paper has outlined an XDD-based framework for support specification and selection in TAPAS, which can formally and uniformly model capabilities and status offered by a running PaP system as well as those required by a play in terms of XDD descriptions. In addition, the framework is equipped with a computation and reasoning mechanism, thus allowing derivation of appropriate play configurations, meeting all the requirements and constraints of a play.

Due to its generality, flexibility and expressiveness, the framework can be extended by a mechanism for retrieval of an optimal (re)configuration, enabling a system to dynamically and automatically fine-tune itself to best handle changing environments. Intuitively, such a mechanism demands abilities to

- formulate play optimisation rules,
- describe, for each role, its resource consumption,
- measure the QoS of the resulting configuration.

Elaboration of such interesting research is underway, and a prototype system demonstrating the efficiency and effectiveness of the proposed framework is developed.

References

1. F. A. Aagesen, B. E. Helvik, U. Johansen and H. Meling. Plug and Play for Telecommunication Functionality: Architecture and Demonstration Issues. *Proc. Int'l Conf. Information Technology for the New Millennium* (*IConIT*), Thammasat University, Bangkok, Thailand, May 2001.
2. J. Hendler and D. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems* 15(2):72–73, Nov./Dec. 2000.
3. Object Management Group. XML Metadata Interchange (XMI) Specification, v1.2, Jan. 2002 [http://www.omg.org/technology/documents/formal/xmi.htm]
4. V. Wuwongse, C. Anutariya, K. Akama and E. Nantajeewarawat. XML Declarative Description (XDD): A Language for the Semantic Web, *IEEE Intelligent Systems*, 16(3):54–65, May/June 2001.

Appendix

XML Declarative Description (*XDD*) [4] is an XML-based knowledge representation, which extends ordinary, well-formed XML elements by incorporation of variables for an enhancement of expressive power and representation of implicit information into so called *XML expressions*. Ordinary XML elements—XML expressions without variable—are called *ground XML expressions*. Every component of an XML expression can contain variables, e.g., its expression or a sequence of sub-expressions (*E-variables*), tag names or attribute names (*N-variables*), strings or literal contents (*S-variables*), pairs of attributes and values (*P-variables*) and some partial structures (*I-variables*). Every variable is prefixed by '$*T*:', where *T* denotes its type; for example, $S:value and $E:expression are *S*- and *E*-variables, which can be specialized into a string or a sequence of XML expressions, respectively.

An *XDD description* is a set of *XML clauses* of the form:

$$H \leftarrow B_1, \ldots, B_m, \beta_1, \ldots, \beta_n,$$

where $m, n \geq 0$, H and the B_i are XML expressions, and each of the β_i is a predefined *XML constraint*—useful for defining a restriction on XML expressions or their components. The XML expression H is called the *head*, the set $\{B_1, \ldots, B_m, \beta_1, \ldots, \beta_n\}$ the *body* of the clause. When the body is empty, such a clause is referred to as an *XML unit clause*, and the symbol '←' will often be omitted; hence, an XML element or document can be mapped directly onto a *ground XML unit clause*.

Intuitively, given an XDD description D, its meaning is the set of all XML elements which are directly described by and are derivable from the unit and non-unit clauses in D, respectively.