An Approach to Capability and Status Modeling

Paramai Supadulchai and Finn Arve Aagesen
Department of Telematics, Norwegian University of Science and Technology (NTNU)
N7491, Trondheim, Norway
{Paramai.Supadulchai, Finn.Arve.Aagesen}@item.ntnu.no

Abstract

A recent trend in network systems is the advanced technology to *dynamically* handle changes in the system. An important basis relies on the integration of *capability and status* representation and their semantic descriptions, which is currently not expressive enough. In this paper, we propose a *Unified Capability and Status Representation Framework (UniCS)* for handling several aspects related to capability and status. A scenario is modeled by UniCS to show how a network system exhibits adaptable behavior.

1 Introduction

Telematics Architecture for Play-based Adaptable System (TAPAS) uses a *Theater Metaphor* to model a network system in the same way a play is played in the theater. Anyone, who has acting skills, can become an *Actor* playing a *Role* in a *Play*. An Actor playing a Role is described as a *Role Figure*. The Role of an Actor is defined in a *Manuscript* containing the behavior of the Role Figure. An Actor is likely to become any Role Figure. However, factors such as sex, age, appearance or acting skill are the obvious barriers that allow some Role Figure to some Actors. These factors are the *Capabilities* of an Actor. It is the job of the *Director* of a Play to determine a fitting Actor to play a Role. In this task, the Director requests help from a *Service Management System*.

In a network system, *Nodes* are typically network processing units such as mobile phone, stationary computer, laptop, printer and router that possess particular *Capabilities*. At a specific time point, a *Status* is a system state with respect to the number of active entities, traffic situation and Quality of Service (QoS) etc. This applies to overall system as well as Nodes. The Capability and Status concepts will be discussed in Section 3.

Nodes have generic *Actors*, which normally inherit Capabilities and Status from the Nodes. The ability of an Actor to play a Role, which is modeled as an *Extended Finite State Machine* in a Manuscript [5], depends on the *Required Capability and Status* of the Role and the *Offered Capability and Status* in a Node where an Actor is going to play [1]. Employing an appropriate *Capability and Status Representation Framework* will help the Director and the Service Management System select the best Actor for a certain Role Figure. However, current Capability and Status representations lack well integration of the *Syntactic Representation* and the *Semantic Description*. This will lead to the difficulty when the Director does not understand clearly the meaning of a Capability or a Status. Though two Actors with slightly different Capabilities can play a specific Role interchangeably, the Director unfortunately does not have such knowledge and thus cannot assign the Role to a working Actor in case that the other one fails.

This paper presents an approach to capture and represent such knowledge in a formal way. Capability and Status are integrated with *Semantic Description* and *Configuration Rules* to enhance the reasoning process. Section 2 gives basic definitions of the *TAPAS Architecture*. An overview of Capability and Status is provided in Section 3. Next, Section 4 describes the definition of the *Play View Capability and Status*. The *Unified Capability and Status Representation Framework (UniCS)* is proposed in Section 5. In Section 6, a scenario demonstrates how to use UniCS to model a Play containing dynamic behavior of an adaptable network system.

2 TAPAS Architecture

The Telematics Architecture for Play-based Adaptable System (TAPAS) intends to be an architecture for *adaptable network systems* that gives *rearrangement flexibility*, *failure robustness*, QoS awareness and *resource control properties* [3]. In analogy with the TINA architecture principles, the architecture is separated in a *Service Architecture* and a *Computing Architecture*. The Service Architecture is an architecture showing the structure of *Services* and *Service Components*. The Service Architecture consists of *Primary Service Providing Functionalities* and additional *Service Systems*. These Service Systems are:

- *Service Management System*: definition of new Services, deployment and invocation of Services and Service Components
- *Capability Management System*: register, de-register, update, transform, provide access to Capabilities and manage Capability ontology.
- *Status Monitoring System*: provision of a view of the offered Status.
- *Configuration Management System*: optimization of Service Systems initial configuration and re-configuration with respect to Capabilities and QoS.
- *Mobility Management System*: The handling of various Mobility types.

The Computing Architecture is a generic architecture for the modeling of any Service Component. The Computing Architecture has three layers: *Service View*, *Play View* and *Network View*. The Service View works seamlessly with the Service Architecture to provide the modeling of an adaptable service. The Play View is the TAPAS specific concepts given in the introduction. The Network View concepts are the basis for implementing the Play View concepts, which again are the basic for implementing the Service View Concepts. In the other way around, the Service View concepts are mapped into the Play View concepts, which again are mapped into the Network View concepts. The Play View concepts are seemingly rearrangement flexibility oriented. The *Capability and Status* concepts, however, give a basis for the further design of the failure robustness, the survivability, the QoS awareness and the resource control properties.

In the Network View, Nodes are installed with the *Core Platform*, which has the execution support for the Play View concepts. Nodes publish their Capabilities and Status through a Capability Management System. However, the Capability Management System is beyond the scope of this paper. This paper covers mainly the representation of Capability and Status, which are managed by the Capability Management System and used by any Service System.

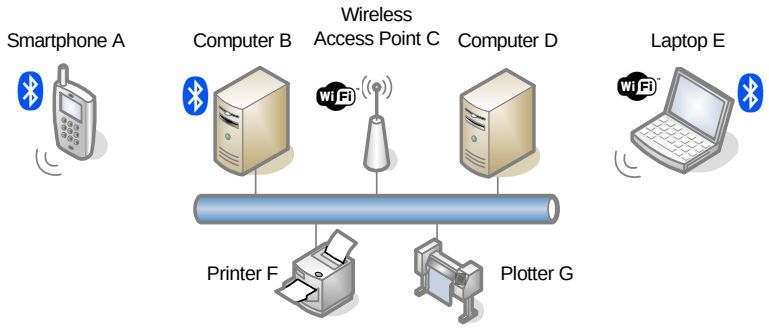


Figure 1: Teleschool environment in TAPAS

3 Capability and Status

We will provide an introduction to Capability and Status by showing the learning facilities of a school illustrated in Figure 1. A classroom is employed with TAPAS to enhance the adaptability of the classroom facilities. From the Figure 1, Node A is the smartphone of a student with a Bluetooth Capability. Laptop E has a Bluetooth Capability and a Wireless 802.11b Capability. These Nodes can connect to the network through Computer B using the Bluetooth Capability. In addition, the Laptop E can connect to the network via Wireless Access Point C.

Computer B has an Ethernet and a Bluetooth Capability. Computer B is installed with a TAPAS Director, a Service Management System and a Capability Management System. Computer D controls the printing jobs of Printer F and Plotter G. All Nodes are installed with the Core Platform and publish their Capabilities and Status through the Capability Management System. Students are assumed to use their own Nodes that have the Wireless Capabilities such as Smartphone A and Laptop E. With these Capabilities, the students can remotely watch lectures live outside the classroom.

Capability

The general definition of Capability in TAPAS is an inherent property of a Node or a User, which defines the ability to do something. Capabilities can be classified as *Resource, Function and Data*:

- Resources: physical processing or storage components or transmission channels with finite capacity
- Functions: pure software or combined software/hardware functions, which perform particular tasks
- Data: just data, which interpretation, validity and life span depend on the context of the usage.

Table 1 shows an example of the Capability list of Computer B. Obviously, physical hardware components are categorized as Resources, which can be independent Nodes or dependent *Hardware Components*. The Hardware Components, such as CPU and memory, are devices that cannot offer functionalities independently and must be hosted by a Node. Functions are softwares that either work internally or provide external interfaces to other Nodes. The examples of the internal function and the external function are Operating Systems and Web Services respectively. Data are just logical information that

is utilized or processed by functions before other Nodes can use. The Data shown in the Table 1 are a username and a password that are accessible from a specific user.

Capability	*Primitives*	*Variety*	*Arrangement*
CPU = Pentium(R) 2GHz	resource	replaceable	shared
Memory = 1 GB	resource	absolute	shared
Disk Storage = 100 GB	resource	absolute	shared
Network Card = 100 Mbit/s	resource	replaceable	shared
Bluetooth = USB 2.0 Bluetooth	resource	absolute	shared
Capabilities = Speaker	resource	replaceable	exclusive
Operating System = Windows XP	function	absolute	exclusive
Web Server = Apache 2.0.46	function	replaceable	shared
Username/Password = john/*****	data	absolute	exclusive

Table 1: The list of capabilities of the device *Computer B* in the three primitive dimensions

A Capability can also be classified in a *Variety Dimension* whether it is *replaceable* or not. If a Capability is replaceable, it can be adjusted or replaced by another Capability with the similar properties. From Table 1, the CPU can be stepped down to save energy when battery is low. In addition, the Web Server can also be equally replaced with a new version when there is a functionality that has been fixed from security threats.

In an *Arrangement Dimension*, a Capability can be shared or exclusive depending on *the visibility and the availability* to other Nodes. For example, the Nodes' CPU information must be visible. This allows the fastest Node to be selected. On the other hand, a password must be hidden. In addition, the Web Server provides a service exclusively to the local computers within the domain when it works in the exclusive mode. Otherwise, it provides the service to all computers in the shared mode.

Status

In addition to the Status definition given in the Introduction, Status and Capability are related in the way that a Capability always has Status but not the other way around. The Status of a Capability is considered the Capability state. A Status can be derived; for example, we can define that a computer is working off-line when either its network adapter is not functioning or the network is not working. This definition is denoted as a Configuration Rule saying that the off-line state of a computer is derived from the states of network and network adapter Capabilities.

Another important aspect about Status is the value range, which is basically of two types: *symbolic or numeric*. The challenge is to provide unambiguous semantic to the value range: for instance, the Connectivity Status of a Node can be defined in only 2 distinct symbolic values: on-line and off-line. The semantic of the off-line state explains that a Node becomes offline when it has lost the connectivity for at least 5 minutes. *Quantifiable Status*, i.e. numerical status, can be defined from three definitions: the semantic of the upper/lower bound, the unit/precision and the impact when the value is changed. An example of the upper/lower bound semantic can be "the more harddisk space a Node has, the more possibility to install a new software". An example of the unit/precision semantic can be "a 100 GB disk is larger than a 100 MB one". An example of the value impact semantic can be "a program uninstallations are needed when harddisk space is used more than 80%".

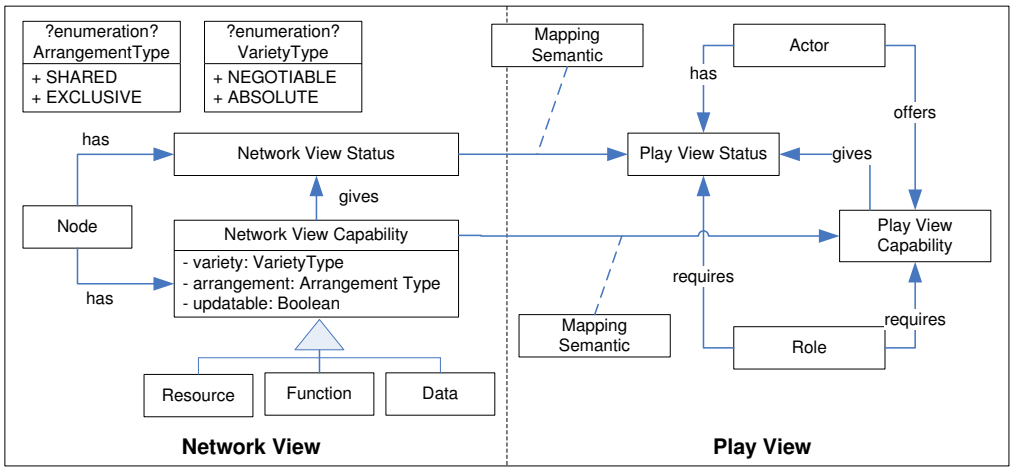


Figure 2: General Capability Model in TAPAS

The Ontological Capability and Status Framework

Capability and Status give extra knowledge to improve the reasoning power of the Director and the Service Management System. However, considering just the solid information about Capability and Status may limit the power of the reasoning mechanism. For example, if a printing system chooses a target printer based on printing-related capabilities, the system may not be able to understand how plotter and printer are different. As a consequence, it would redirect a billing report to a costly plotter. Thus, it is not sufficient to define that two printers can be used interchangeably when they supply the same printing-related capabilities. Some constraints and taxonomy hierarchy are needed; for examples: "The cost-per-page difference of both printers must not exceed 10%." or "The plotter is categorized as high-resolution printer and the high-resolution printer is available exclusively to a large document.". These requirements constitute *Configuration Rules*, which can be used to make a common understanding of a specific Capability and Status. Configuration Rules also provide the relationship between well-defined Capabilities and Status. In this way, an *Ontological Capability and Status Framework* can be created.

4 The Play View Capability and Status

To properly perform the reasoning mechanism in adaptable networks, the proper understanding of Capability and Status is needed. A system designer can compose a *Play* to fulfill this requirement by combining several Configuration Rules that consist of *Capabilities*, *Status*, *Axioms* and *Constraints*. However, the system designer would have to design the Play that covers all possible Capabilities and Status of any Node. Alternatively, we propose the use of an abstraction layer of Capability and Status for the Play composition. Capability and Status are considered in the Network View and the Play View as illustrated in Figure 2. What up to now that has been considered as Capabilities and Status of a Node are denoted as Network View Capabilities and Status, abbreviated as NV-Capabilities and -Status respectively.

A Play View Capability, abbreviated as PV-Capability, is the function that an NV-Capability offers in the Play View. It is an *abstraction* of an NV-Capability that

1. provides a unified representation with well-defined semantic to describe a Capability in the Play View,

2. maps two or more NV-Capabilities that potentially give the same functionality,

3. simplifies the *Play* creation process from the direct use of NV-Capabilities,

4. has specific meaning in a Domain.

Let's consider the Teleschool environment given in the Figure 1. Here we encounter two mobile Nodes with dissimilar Wireless NV-Capabilities. The *Smartphone A* uses Bluetooth to connect to other Nodes, while the *Laptop E* connects to the network with its 802.11b Wi-Fi. If the Director wants to execute a Play to detect all Nodes with Bluetooth or 802.11b NV-Capability, the Director will have to find out these Capabilities iteratively from all Nodes. If a student owns a laptop with a newer version of Wi-Fi 802.11g, the new NV-Capability *Wi-Fi 802.11g* must be included in the Play. Otherwise, the Play is not applicable to all possible Nodes.

Wouldn't it be nice if we can define shortly a Configuration Rule in the play "all Actors possessing wireless PV-Capability must turn off their Sound PV-Capability before entering a Domain that is offering classroom PV-Capability."? With the assumption that all NV-Capabilities can be mapped to PV-Capabilities, the play can be designed in a simpler manner from the PV-Capability requirement instead of using the NV-Capability directly. PV-Capabilities can be categorized into an ontology hierarchy; for example: wireless PV-Capability can be further separated to the long-range, mid-range and short-range transmission functions. The PV- and NV-concept are also applicable to Status the same way it is used for Capability. However, examples will not be given.

5 Unified Capability and Status Representation Framework (UniCS)

From earlier examples, we have given the examples of the Configuration Rule, which are used to formulate a Play. However, these rules are not meant to be performed by us. They shall be enforced by the network system. Therefore, a formal way to capture and make use of the Configuration Rules is required. In this section, a proposal for the formal representation of the NV-Capability and -Status and the PV-Capability and -Status is given in a unified framework denoted as the Unified Capability and Status Representation (UniCS).

Network View Capability and Status Representation

The Network View Capability and Status, abbreviated as NV-Capability and -Status, are used to describe Capability and Status extracted from Nodes. Their representation should provide a well-defined structure with precise semantic understandable by the network systems. Though any representation can be used with the NV-Capability and NV-Status because they will be eventually transformed into the PV-Capability and -Status, using a single standard accepted by all Domains is recommended. With a single standard representation, the system does not have to worry about how to read and understand the semantic of the representation since it is what everybody has agreed on. There are and will be a lot of ready-to-use tools and supports for the standard representation. In addition, only one set of Configuration Rules is required to map standardized NV-Capabilities and -Status into PV-Capabilities and -Status. Using a single representation therefore simplifies the integration of the instances of Capabilities and Status from many Domains.

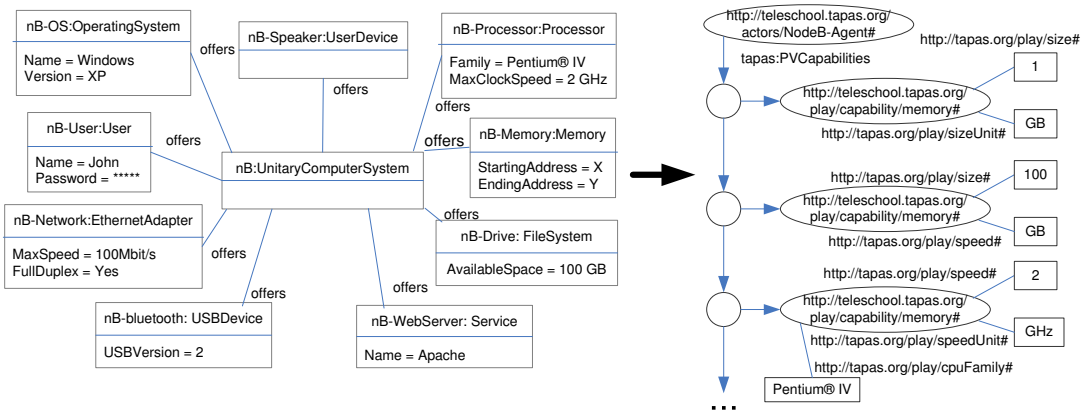


Figure 3: The transformation of NV-Capabilities and -Status of Computer B to PV-Capabilities and Status

One of the emerging standards in networking data model is *Common Information Model (CIM)* from Distributed Management Task Force (DMTF). The standard is based on the attempt to integrate all management information. On the basis of UML used in the model, CIM is an easy-to-use model with a lot of graphical modeling tools and Application Programming Interfaces (APIs) on many computing platforms. The model comes with sufficient pre-defined physical, logical and service entities with the possibility for extensions. The CIM serialization in the Extensible Markup Language (XML-CIM) inherits an important property of XML, the machine comprehensible. The example of modeling the NV-Capability and -Status in CIM is illustrated in Figure 3.

CIM provides only the basic semantic to each entity in the model. To cope with this limitation, we develop a framework that improves the expressiveness of CIM models by constructing XML Declarative Description (XDD)-based Configuration Rules with axioms, constraints and a reasoning mechanism to facilitate additional semantic to the model entities. XDD [11] is an expressive XML-based knowledge representation that works seamlessly with all models, syntax and ontologies by extending ordinary well-formed XML ground element with variables. Aagesen et al. gave an example of a dynamic configurable system in the Network View with CIM and XDD [2].

Play View Capability and Status Representation

In the Play View, an Actor inherits Capabilities and Status from a Node. These are Play View Capabilities and Status, abbreviated as PV-Capability and -Status. Unlike the NV-Capability and Status, the PV-Capability and Status are represented in a unified and platform-independent representation: *Resource Description Framework* (RDF). Each PV-Capability is an RDF resource with a *Uniformed Resource Identifier (URI)* that belongs specifically to a domain. An RDF statement is composed by *a triple consisting Subject, Verb, Object* that resembles an English sentence. Even though the RDF structure is simple, it provides sufficient constructs to describe PV-Capabilities and -Status. In addition, the standard ontology languages on the web such as the RDF Schema (RDFS) and the Web Ontology Language (OWL) are also based on RDF. This enhances the reasoning mechanism by the creation, manipulation and exchange of the ontology of the PV-Capability and -Status across Domains. Anutariya et al. proposed an RDF Declarative Description (RDD) framework, which shows the effectiveness of using the ontological language such as RDF, DAML+OIL and DAML-S with the Configuration Rules described

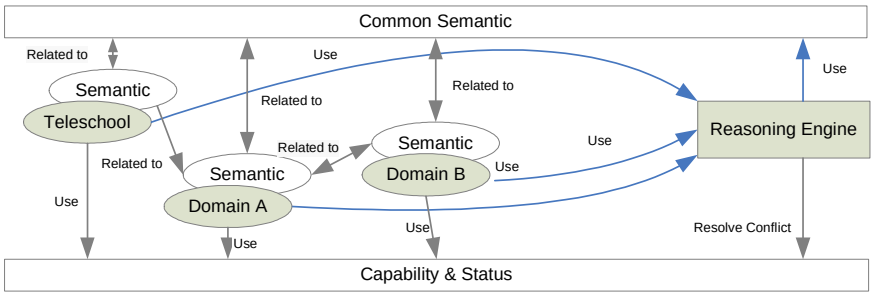


Figure 4: UniCS's ontological Capability and Status handling

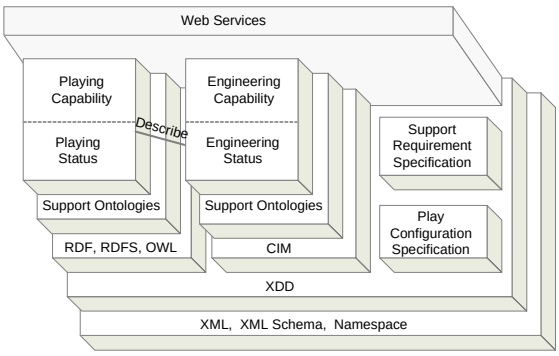


Figure 5: UniCS's Capability and Status Representation Layer (extended from [1])

in XDD. [4]. The RDD framework is the basic idea of employing PV-Capability and PV-Status in TAPAS.

Figure 3 shows the transformation from the Computer B's NV-Capability represented in CIM instance to the equivalent PV-Capability in RDF. Due to the space limitation, the complete RDF diagram is not given.

Domain Oriented Representation

In TAPAS, a *Domain*, taken care by a single Director, is a part of a big system containing Nodes that are related in terms of location, functionality or service. The Internet is a good example of a Domain System. Computers on the Internet are divided into groups with unique domain names. UniCS is a domain-oriented representation, i.e. the semantic of Capability and the Status may vary from place to place, from time to time or from a specific situation to others. Since it is not possible to force a single semantic framework to all systems, a *Common Semantic* is used to resolve semantic conflicts in different Domains.

In Figure 4, each Domain is supplied with its own semantic with the relationship to the Common Semantic. The Common Semantic provides the relationship between identical Capabilities and Status from various Domains. It is used to resolve possible conflicts from Capabilities and Status with different semantic and thus make it possible to combine them. For example, the Smartphone concept is defined in a Domain as a mobile phone that requires Microsoft Windows Mobile Edition for Smartphone. This is because the Domain can only supply support software for this platform only. However, the case might not be the case in some Domains where the Smartphone Operating System can vary from Symbian, Microsoft Windows Mobile Edition or Linux.

The concept of Domain applies to both NV-Capability and -Status and PV-Capability and -Status. Nonetheless, UniCS handles the NV-Capability and -Status differently.

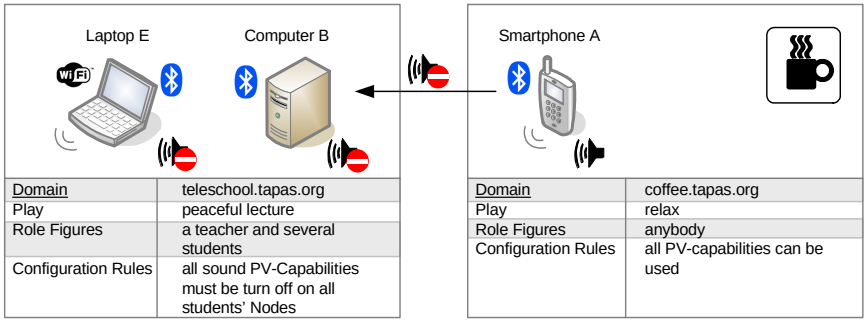


Figure 6: The example of an adaptable network system in a network classroom

The semantic of the NV-Capability or an NV-Status is mapped equivalently to one or more PV-Capabilities or -Status defined in the Domain. This reduces the complexity of handling the semantic differences of NV-Capabilities and -Status across domains, and thus provides a unified framework to the Capability and Status Management by using the PV-Capability and -Status. In addition, UniCS allows the flexible interoperability with other architectures that encode Capability and Status in different syntax and semantic. With Configuration Rules supplied, the Capability and Status representations in those architectures can be transformed equivalently into the UniCS's PV-Capability and -Status.

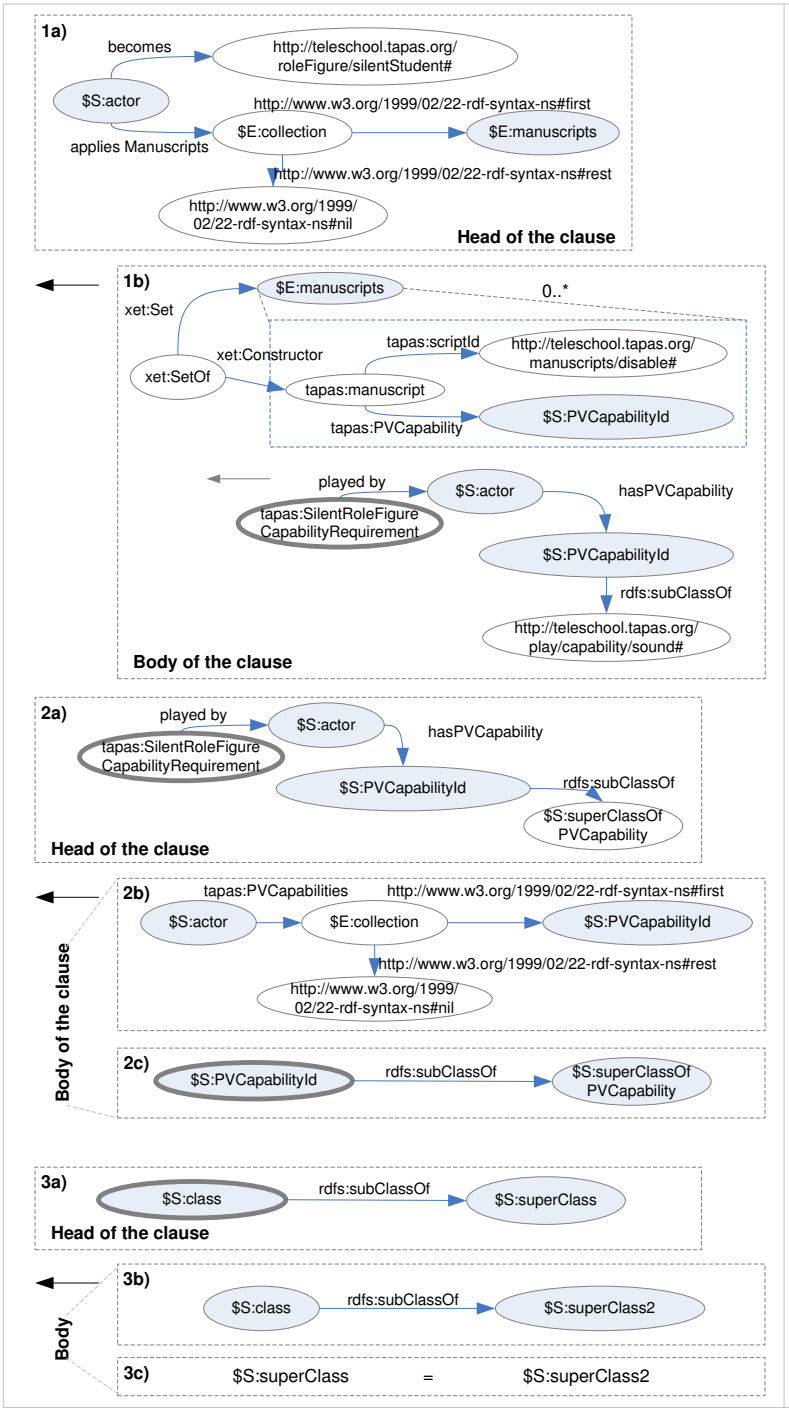
6 Using Play View Capability and Status

We have briefly shown how Capability and Status in TAPAS are represented in UniCS. Basically, UniCS separates NV-Capability and -Status from PV-Capability and -Status. Configuration Rules, which constitutes the Play in TAPAS, are created from PV-Capabilities and -Status. The representation layer showing the representation used in UniCS is illustrated in Figure 5. We are now ready to use the PV-Capability and -Status to create Configuration Rules for a scenario in the teleschool environment.

Figure 6 reveals the facilities of a network classroom. Since it has been recently reported that many students don't turn off their mobile phones when they are in the classroom, the teleschool administrator is asked to construct a play in the TAPAS environment to automatically detect and request the smartphone Nodes to turn off the *Sound PV-Capability*. Fortunately, the students' smartphones are installed with the TAPAS Core Platform that provides support for the Play-based concepts.

It is assumed that all PV-Capabilities that produce sound must be turned off separately. Therefore, the system administrator must create a Play to verify all PV-Capabilities of the Actors hosted in the students' Nodes, i.e. the smartphones. If Nodes have PV-Capabilities that are the *subclasses* of the *Sound PV-Capability*, the Director will send a *"Disable Manuscript"* to turn them off. Based on UniCS, three Configuration Rules are created as illustrated in Figure 7.

When the smartphone A, with an MP3player, a Loudspeaker and a Bluetooth PV-Capabilities, enters the classroom, two PV-Capabilities, the Loudspeaker and the MP3player, will be turned off. This is because they are the subclasses of the sound PV-Capability. With the provided subclass-superclass ontology, the Director create a *Play Configuration* instructing an Actor in the smartphone A, smartphoneA-Agent, to turn off these PV-Capabilities with a Manuscript. The execution result is shown in Figure 8.



Rule 1 – Determines Manuscripts required when a new Actor wants to play the *Silent Student Role*.

1a) Any instances of the Actor can become the *Silent Student Role Figure*. However, they must execute a collection of Manuscripts.

1b) A list of Manuscripts will be constructed. The *xet:SetOf* constructs the list of Manuscript *$E:manuscripts* based from the constructor specified within *xet:Constructor*.

A Manuscript is the disable script (specified by the script id). The parameter of the script is the name of the PV-Capability to be disabled.

The xet:SetOf function will be fired exclusively for *all PV-Capabilities* of the Actor $S:actor that are subclasses of the Sound PV-Capability. This implicitly ignores all irrelevant PV-Capabilities

Rule 2 – Find out *the PV-Capabilities* of *an Actor* that are *subclasses* of *a PV-Capability*.

2a) Any Actors *$S:actor* can have PV-Capabilities that are subclasses of a PV-Capability *$S:superClassOfPVCapability*

2b) This body represents the structure how an Actor *$S:actor* possesses the PV-Capabilities (in the RDF collection model).

2C) The restriction stated in this body shows that all applicable PV-Capabilities must be the subclasses of *$S:superClassOfPVCapability*. The restriction will be proved by the third rule.

Rule 3 – Prove if a resource is the subclass of another resource

3A) A resource $S:class can be the subclass of a resource *$S:superClass*.

3B) If for all instances of resource *$S:class* are a subclass of another resource *$S:superClass2*.

3C) And $S:superClass2 is the same resource as $S:superClass

Figure 7: The Play to turn-off all kinds of PV-Capabilities that produce sound.

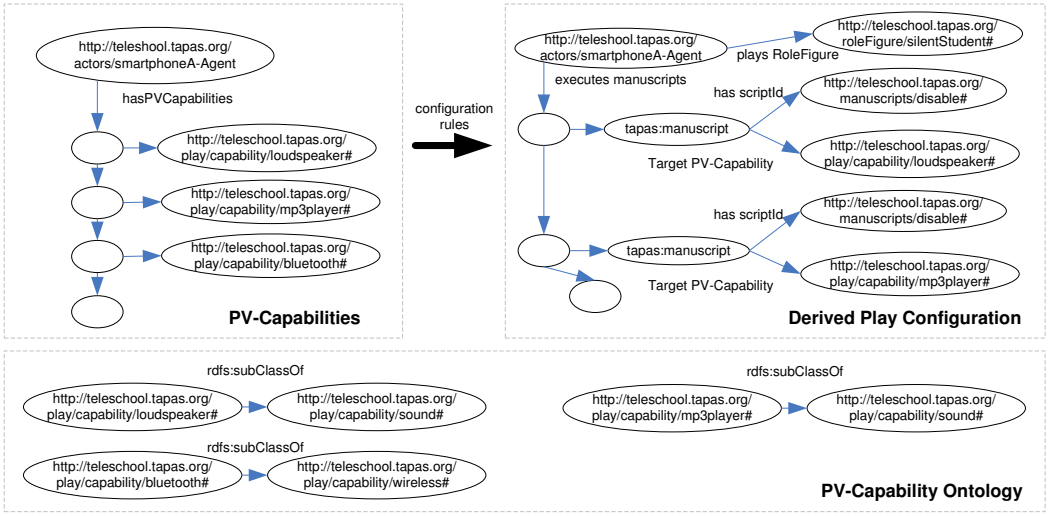


Figure 8: The execution result of the Play in Figure 7 and the PV-Capabilities with supplied ontologies.

7 Conclusion

By using UniCS's PV-Capability and -Status, the semantic of the Capability and Status models of network systems can be enriched. As a result, a play-based network system, led by a Director, can orchestrate itself to behave properly when there are changes occurring. This reasoning mechanism serves as a principle of the *rearrangement flexibility* that is one of the three basic required properties for the *adaptable system*. Nevertheless, there are still more works to do regarding the rearrangement flexibility since the simple subclass-superclass relation is not expressive enough in all situations. We are working on the next version of UniCS to provide more ontological ingredients. The future integration with the RDD framework will enlarge the scope of the ontological framework in UniCS, and thus will make the ontological model suitable in most situations. In addition, the model must be combined with other works regarding failure Robustness and resource load awareness and Control in TAPAS. The final model will then serve as the true grounding for the Adaptable System.

8 Related Works

CIM is used in many projects to resolve the interoperability problems between systems. Examples are the Web-Based Enterprise Management (WBEM) and the Directory Enabled Networks (DEN) from DMTF [10]. However, the frameworks lack a mechanism to enable the dynamic behavior. The Automated Policy-based Resource Construction by Sahai et al. employs CIM as the underlying Capability model [8]. Nevertheless, the policies are constructed mainly by the defined constraints. The lack of expressive axioms limits the system from deriving new knowledge, which is unfortunately needed in adaptable environments.

There are several attempts trying to provide a unified framework to model syntax and semantics using XML and RDF. This is again emphasized by Tim Burners-Lee when he encouraged developers to start building RDF triples that contain ad-hoc ontologies in the WWW2004 conference. Patel-Schneider and Siméon employ an RDF mediator to allow XML dialects in the applications [7]. Although this is quite relevant to our work, their research focuses on providing the semantic reasoning to static models. Thus, it is

not well-suited in the world of network application. The RDF-based system can also be found in the world of network management. Shen and Yang use the RDF to describe models created by the next generation structure of management information (SMIng) [9]. However, the work tends to focus mainly on the resource level, not Capability. Another attempt by Motik and Glavinić to create model for querying RDF knowledge in the Agent Architecture is limited to the RDF and not applicable to other ontological languages [6].

References

[1] Finn Arve Aagesen, Chutiporn Anutariya, Mazen Malek Shiaa, and Bjarne E. Helvik. Support specification and selection in tapas. In *Proceedings of IFIP WG6.7 Workshop and Eunice Summer School on Adaptable Networks and Teleservices*, September 2002.

[2] Finn Arve Aagesen, Chutiporn Anutariya, Mazen Malek Shiaa, Bjarne E. Helvik, and Paramai Supadulchai. A dynamic configuration architecture. In *IEEE/IFIP Network Operations and Management Symposium NOMS'2004*, Seoul, South Korea, April 2004.

[3] Finn Arve Aagesen, Bjarne E. Helvik, Chutiporn Anutariya, and Mazen Malek Shiaa. On adaptable networking. In *Proceedings of the First International Conference on Information and Communication Technologies, ICT'2003*, Assumption University, Thailand, April 2003.

[4] Chutiporn Anutariya, Vilas Wuwongse, Kiyoshi Akama, and Ekawit Nantajeewarawat. Rdf declarative description (rdd): A language for metadata. *Journal of Digital Information*, 2(2), 2001.

[5] Shanshan Jiang and Finn Arve Aagesen. Xml-based dynamic service behaviour representation. In *NIK'2003*, Oslo, Norway, November 2003.

[6] Boris Motik and Vlado Glavinić. Enabling agent architecture through an rdf query and inference engine. In *In Proceedings of the 10th Mediterranean Electrotechnical Conference, MEleCon 2000*, volume 2, pages 762–765, Cyprus, 2000.

[7] Peter F. Patel-Schneider and Jérôme Siméon. The yin/yang web: A unified model for xml syntax and rdf semantics. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):797–811, July/August 2003.

[8] Akhil Sahai, Sharad Singhal, Rajeev Joshi, and Vijay Machiraju. Automated policy-based resource construction in utility computing environments. In *IEEE/IFIP Network Operations and Management Symposium NOMS'2004*, Seoul, South Korea, April 2004.

[9] Jun Shen and Yun Yang. Rdf-based knowledge model for network management. In *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003)*, Colorado Springs, CO, USA, March 2003. Kluwer Academic Publishers.

[10] Andrea Westerinen and Winston Bumpus. The continuing evolution of distributed systems management. *IEICE TRANS. INF & SYST.*, E86-D(11):2256–2261, November 2003.

[11] Vilas Wuwongse, Chutiporn Anutariya, Kiyoshi Akama, and Ekawit Nantajeewarawat. Xml declarative description (xdd): A language for the semantic web. *IEEE Intelligent Systems*, 16(3):54–65, May/June 2001.