2010 Fifth International Multi-conference on Computing in the Global Information Technology

# Capability Ontology in Adaptable Service System Framework

Patcharee Thongtra
Department of Telematics
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
patt@item.ntnu.no

Finn Arve Aagesen
Department of Telematics
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
finnarve@item.ntnu.no

***Abstract*—This paper presents a Capability Ontology (CapOnt) and a rule-based reasoning mechanism, which support service management within adaptable service systems. The ontology concepts comprise capability types, capability parameters and service management functions related to capabilities. Capability parameter values can be defined by constraints on other capability parameters. The service management functions included in the ontology are Capability Administration, Capability Configuration, Capability Allocation and Capability Performance Diagnosis. The service management functions are defined by rules consisting of constraints and management actions. The ontology concepts are represented in OWL (Web Ontology Language) and OWL/XDD (XML Declarative Description Language) – a rule-oriented knowledge representation. An intelligent conference room example with simulation results, that demonstrates the CapOnt and the rule-based reasoning, is also presented.**

***Keywords-Capability; Network and service management; Adaptable system; Ontology***

## I. INTRODUCTION

In this paper, networked services are considered. A service system consists of inter-working service components. An adaptable service system is here defined as a service system that can adapt dynamically to changes to service users, nodes, capabilities, system performance and service functionality. *A capability is an inherent property of a node to implement service [1]*. Capabilities can be classified according to capability types and capability parameters. A network interface with finite bandwidth is an example of a capability type. A service component needs capabilities to be allocated before deployment and instantiation. The capabilities can be re-allocated in situations when the system performance is not satisfactory. Formal concepts are, therefore, needed as a basis for the administration, configuration, allocation, monitoring, and performance diagnosis of capabilities.

The software mechanisms used for implementing the functionality of adaptable service systems must be flexible and powerful. Service components based on the classical Extended Finite State Machine (EFSM) approach can be flexibly executed by using generic software components that can download and execute different EFSM-based specifications. In addition, combining *rule-based reasoning* with the EFSM-based approach adds the ability to cope with various situations more flexible. The rule-based reasoning is considered as support functionality for the EFSM-based service components; it suggests actions to the service components.

Considering the capability concepts, there are some standard information models comprising defined capabilities, e.g. SNMP's Structure of Management Information (SMI) [2] and Common Information Model (CIM) [3]. However, they are limited in terms of using rule-based concepts. In this paper, we propose an ontological approach to model and represent the capability concepts including rule-based concepts.

An ontology is a formal and explicit specification of a *shared* conceptualization [4]. An ontology, in general, consists of types, properties, instances, relations and rules. The rule can place constraints on sets of types, properties and relations allowed between them. *Capability Ontology (CapOnt)* has capability types, parameters, instances, relations and service management functions. The parameters are the properties and inference relations of the capability types. The parameter values can be defined by constraints on other parameters. The service management functions, which are capability-related actions to fulfill the service management functionalities, are also included in this ontology. This is because the actions are shared concepts among distributed service components which implement the service management functionalities. The service management functions are defined by rules consisting of constraints and the actions.

In principle, all network management objects, operations on these objects, and network and service management functionality can be defined in ontologies. In this paper, our ontology defines capability types, parameters and service management functions.

The main contributions of this paper are:

- A service functionality architecture for adaptable service systems.
- A Capability Ontology (CapOnt) that includes rule-based parameters and service management functions.
- A rule-based reasoning procedure. The reasoning procedure is implemented by Native xml Equivalent Transformation reasoning engine (NxET) [5].

The rest of the paper is organized as follows. Section II reviews related works. Section III presents the service functionality architecture. Section IV presents the Capability Ontology (CapOnt) concepts and representation. Section V

978-0-7695-4181-5/10 $26.00 © 2010 IEEE
DOI 10.1109/ICCGI.2010.14

explains the reasoning procedure. In section VI, an example of intelligent conference room is given. Finally, section VII presents summary and conclusions.

## II. Related Work

### A. Standardized Information Models

The IETF proposed Structure of Management Information (SMI) [2] to be used in the SNMP framework [6], followed by Next Generation Structure of Management Information (SMIng) [7]. SMIng provides mechanisms to formally specify constraints between values of multiple parameters, though its implementation was not accomplished. In the other models: Guidelines for the Definition of Managed Objects (GDMO) [8] and Common Information Model (CIM) [3], the object-oriented modeling approach is applied. GDMO has been widely used in telecommunications networks. CIM provides the definitions for systems, networks, users and services. It was proposed with the intention to use in the Web-based management architecture. However, we can not easily define rule-based managed objects, which their values are restricted by other managed objects, by these information models alone. SNMP RMON MIB allows that to some extent for statistics gathering, but it is complicated. López de Vergara et al. [9] used OCL as an extension model to write rules in the CIM metaschema.

### B. The Application of Ontology

There are various directions for the application of ontology. In [10]-[12], the ontology-based mapping techniques are used to integrate existing information models. In [13]-[15], the formal definitions of concepts including rules by the ontology-based approaches were presented. Guerrero et al. [13] used OWL [16] and SWRL to define constraints on the parameter values of the SNMP- and the CIM managed objects. They also modeled behaviour rules composed of the actions and constraints. The actions are general functions, that are performed by the manager in the traditional manager-agent framework when the constraints are met. In [14], the behaviour rules were defined with a focus on the network configuration management. Diaz et al. [15] mapped the original configuration of wireless routers into the CIM schema, and later translated these into OWL. They further added constraints on the relations allowed between the managed objects, with which to detect router configuration errors. The approach used in this paper is similar to [13],[14]. But the Capability Ontology has an application domain different from the others. The capability concepts are defined towards the service management functionalities for adaptable service systems.

## III. Service Functionality Architecture

Service functionality architecture as illustrated in Fig. 1 defines the structure and content of service functionalities independent of implementation. The architecture is intended to provide three classes of the adaptability: *rearrangement flexibility*, *failure robustness*, and *QoS awareness and resource control*. *Rearrangement flexibility* means that the service system structure and functionality are not fixed. Nodes, users, services, service components, capabilities, can be added, moved, removed according to needs. Mobility of users, sessions, nodes, terminals is further seamlessly handled. *Failure robustness* means that the architecture is dependable and distributed, and that the system can reconfigure itself in the presence of failures. *QoS awareness and resource control* means that there is functionality for negotiation about QoS and optimum resource allocation, monitoring of resource utilization, and actions for reallocation of resources. Fundamental QoS concepts considered are capability, capability performance, service performance and service level agreements (SLAs).

The functionalities of the service functionality architecture are classified as *primary service functionalities* and *service management functionalities*. The *primary service functionalities* provide services to the service users. The *service management functionalities* consist of functionality components and repositories to manage services as well as nodes and their capabilities. The functionality components and repositories will be explained later in this section.

The service management functionalities and the primary service functionalities can be implemented by Extended Finite State Machine (EFSM)-based and Reasoning Machine (RM)-based service components [17]. In this paper, the RM-based service component is used as the traditional procedure that provides the rule-based reasoning to the EFSM-based service component.

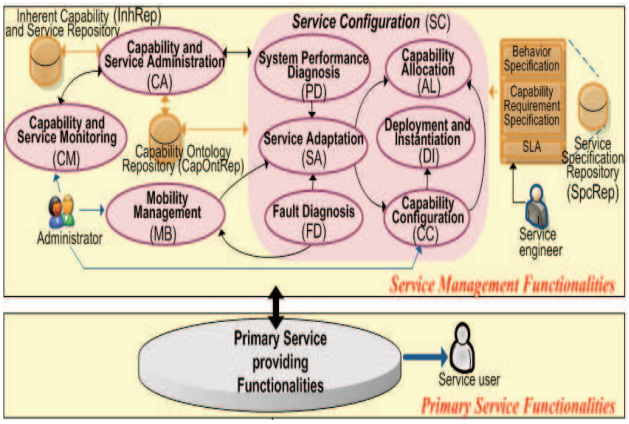



Figure 1. Service Functionality Architecture.

- *Capability Ontology Repository (CapOntRep)* stores the capability concepts.
- *Service Specification Repository (SpcRep)* stores the service component behavior specifications, the capability requirements, and the SLAs. The capability requirements define the required capability types and parameters for the various service components.
- *Inherent Capability and Service Repository (InhRep)* stores data about available nodes, the inherent capabilities and instantiated services. The inherent capabilities are the existing instances of capability types and parameters for the various nodes.

- *Capability and Service Administration (CA)* handles the registration of nodes, the inherent capabilities and instantiated services. CA also provides a current view of available nodes, the inherent capabilities and instantiated services.
- *Capability and Service Monitoring (CM)* monitors node, their capabilities and instantiated services. CM gets the monitoring requests from the administrator as well as CA, and updates the monitored data to CA.
- *Mobility Management (MB)* handles various mobility types such as personal mobility, terminal mobility and service component mobility.
- *Service Configuration (SC)* has the sub-components as illustrated in the figure. *Capability Configuration (CC)* finds and selects capable nodes that satisfy the capability requirements. The *capable node* has capabilities as same as or as compatible as ones specified in the requirements. *Capability Allocation (AL)* allocates capabilities to service components in various service classes. The task is performed in accordance with the requirements of system performance, here defined as the sum of service performance and capability performance, in the SLAs. *Deployment and Instantiation (DI)* comprises *deployment* which is the introduction of new service components in nodes, and *instantiation* which starts execution of deployed service components. *Fault Diagnosis (FD)* detects the failure of the instantiated services. *System Performance Diagnosis (PD)* detects mismatch between the required system performance and the inherent system performance. *Service Adaptation (SA)* plans the adaptation during the service system execution in case of 1) mismatch between the required system performance and the inherent system performance, 2) faults and 3) other reasons for service component movements. The adaptation results in AL only, or the combination of CC and AL.

In this paper, the capability is focused on. Accordingly, CA and PD are simplified to Capability Administration and Capability Performance Diagnosis. AL considers only the required capability performance.

## IV. CAPABILITY ONTOLOGY

### A. Concepts

The concepts of the Capability Ontology (CapOnt) are illustrated in Fig. 2. *Capability type* defines entities with common characteristics. A capability type can be functions, physical resources and data. The function examples are operating system, application software and protocol. The physical resource examples are CPU and memory. User accounts and passwords are the data examples.

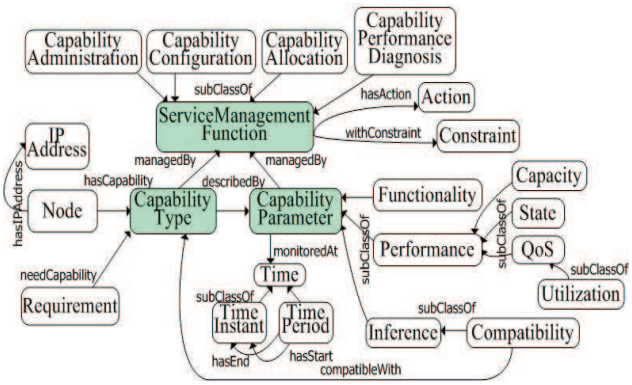



Figure 2. General Concepts of Capability Ontology.

*Capability parameter* describes the characteristics of a capability. *Functionality* parameters define features of the functionality and *performance* parameters describe features of the performance. *Inference* parameters define relations to other capability types, in which these relations are logically concluded from other parameters. The functionality and the performance parameters *can* be rule-based, while the inference parameters *are* rule-based.

*Performance* parameters are further classified into *capacity*-, *state*- and *QoS* parameters. *Capacity* parameter examples are transmission channel capacity, CPU processing speed and disc size. *State* parameters define the situation of the capability at a specific time. State parameter examples are the number of available streaming connections and number of packets discarded in a specific buffer. *QoS* parameter defines the degree of satisfaction of the service users. QoS parameters can be traffic and dependability statistics based on observed stochastic variables. Important traffic variable examples are throughput and utilization. Important dependability variable examples are availability and recovery time.

The parameter *Utilization* and *Compatibility* are important rule-based parameters used in the example in Section VI. *Utilization* is a QoS parameter that gives the average usage of a capacity over a defined time interval. *Compatibility* is an inference parameter that specifies a relation from a capability to other compatible capabilities, in which these capabilities can be used as its substitutes.

The service management functions are defined by rules as mentioned already in Section I. The service management functions are classified into Capability Administration, Capability Configuration, Capability Allocation and Capability Performance Diagnosis, which corresponds to the functionality components presented in Section III.

The CapOnt consists of general concepts of capabilities. For a complete capability specification of a specific system, a *System-specific Capability Ontology (SysCapOnt)* must be defined. The SysCapOnt inherits the CapOnt. The service management functionalities presented in Section III will need instances of the capability types and the capability parameters part of the SysCapOnt for CA, CC, AL and PD.

### B. Representation

OWL [16] and OWL/XDD [18] are used to represent the capability concepts. OWL is a standard Web ontology language, which provides a rich set of constructors. The capability types, the non-rule-based parameters, their relations and instances are expressed by *owl:Class*, *owl:Class*, *owl:ObjectProperty* and *OWL instances* respectively. The values of non-rule-based parameters are specified by *owl:DatatypeProperty* that is called *hasValue*.

However, OWL is limited when it comes to describe the rule-based capability concepts. OWL/XDD, a rule-oriented knowledge representation, is then required to express such concepts. OWL/XDD extends ordinary XML-based elements by incorporation of *variables* for an enhancement of expressive power and representation of implicit information into so called *XML expressions*. The ordinary XML-based elements – XML expressions without variables – are called *ground XML expressions*. The variables are classified into five types as explained in Table I. A variable is prefixed by '$T:' where T denotes its type.

TABLE I. THE VARIABLE TYPES

| Variable type | Pre fix | Instantiated to |
|---|---|---|
| N-variable | $N: | An element tag name or an attribute name |
| S-variable | $S: | An element value or an attribute value |
| P-variable | $P: | A sequence of zero or more attribute-value pairs |
| E-variable | $E: | A sequence of zero or more XML expressions |
| I-variable | $I: | Part of XML expressions |

A rule-based parameter and a service management function are represented as an XML clause of the form:

$$H \rightarrow B_1, .., B_m, \{C_1, .., C_n\} \quad (1)$$

where $m, n \geq 0$, $H$ and $B_i$ are XML expressions, and each of $C_i$ is a pre-defined XML condition on the XML clause. $H$ is called the *head* of the clause, while the set of $B_i$ and $C_i$ is the *body* of the clause. When the body is empty, such a clause is referred to as an *XML unit clause* and the symbol '$\rightarrow$' will be omitted. Hence an XML-based element or document can be mapped directly onto a ground XML unit clause.

## V. REASONING PROCEDURE

The reasoning procedure begins with an XML expression based query $Q$. Then, an XML clause, called query clause, is formulated from the XML expression as follows:

$$Q \rightarrow Q \quad (2)$$

The XML expression $Q$ represents the constructor of the expected answer which can be derived if all conditions in the body of the clause hold. However, if one or more XML expression bodies still contain the variables, these variables must be matched and resolved from other unit clauses and non-unit clauses.

A body from the query clause will be matched with the head of other clauses. At the beginning, there is only one body $Q$. Consider a clause $R_1$ in the form:

$$R_1: \quad H \rightarrow B_1, B_2, C_1 \quad (3)$$

If the XML structure of the body $Q$ and the head $H$ of the clause $R_1$ match without violating the condition $C_1$, the body $Q$ will be transformed into $B_1$ and $B_2$. All variables in the head $Q$ and the new bodies $B_1$ and $B_2$ of the query clause will be instantiated. The query clause will be in the form:

$$Q^* \rightarrow B_1^*, B_2^* \quad (4)$$

where $X^*$ means the one or more variables in the XML expression $X$ has been instantiated and removed.

The transformation ends when either 1) the query clause has been transformed into a unit clause or 2) there is no clause $R_x$ that can transform the current bodies of the query clause. If the constructor $Q$ is transformed successfully into $Q_f$ that contains no variable, the reasoning procedure ends and the desired answered is obtained.

## VI. CASE STUDY

This section describes an intelligent academic conference room example. A *conference service system* provides video transfer services, and users can download and watch present presentations of accepted papers, present tutorials, and old videos of the presentations and tutorials from previous years. The entire video files will be transferred to the user devices before they can start. In this scenario, the services are classified by the video types: the present presentation, the present tutorial, and the old video. The old video has low priority. The users access the services from their wireless communication devices. However, when the number of users increases, the bandwidth usage becomes too high. When this occurs, the services should be degraded gracefully. This means only high priority services will be available, but they are operating regularly.

The conference service system has adaptability features that deploy and instantiate video transfer service components in capable nodes. Also, it disables the old video transfer services when the wireless bandwidth utilization is greater than a maximum limit, and enables them when the bandwidth utilization is less than a minimum limit.

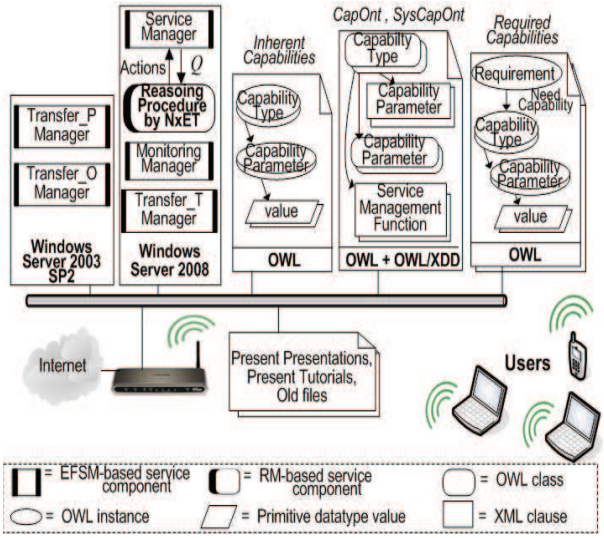



Figure 3. The conference service system.

The conference service system consists of EFSM-based service components and an RM-based service component in nodes as illustrated in Fig. 3. The EFSM-based service components are *Service Manager*, *Monitoring Manager*, *Transfer_P Manager*, *Transfer_T Manager* and *Transfer_O Manager*. A System-specific Capability Ontology (SysCapOnt) will be presented in Section VI.A. With reference to the service functionality architecture in Section III, the Service Manager implements part of Capability Administration, Capability Configuration, Deployment and Instantiation, Capability Performance Diagnosis, Capability Allocation and Service Adaptation. The Monitoring Manager implements part of Capability and Service Monitoring, and the RM-based service component is the reasoning procedure used by the Service Manager. The Transfer_P Manager, Transfer_T Manager and Transfer_O Manager offer the transfer services for the following video types respectively: the present presentation, the present tutorial, and the old video.

The Service Manager and Monitoring Manager are instantiated by the administrator. These managers play an important role to attain the adaptability features as mentioned above. The Service Manager has sub functions as follows:

*a)* Discover available nodes within the conference room.

*b)* Call the reasoning procedure and get an action: request the Monitoring Manager to monitor a set of capability types and parameters: *{$S:CapabilityType, {$S:CapabilityParameter}}*, from the nodes every interval.

*c)* Call the reasoning procedure and get an action: select capable nodes: *{$S:Node}*, that satisfy the capability requirements of the Transfer_P Manager, Transfer_T Manager and Transfer_O Manager.

*d)* Deploy and instantiate the Transfer_P Manager, Transfer_T Manager and Transfer_O Manager in the capable nodes.

*e)* Get the monitored capability instances: *{$S:CapabilityTypeInstance, {$S:CapabilityParameterInstance, $S:value}}*, from the Monitoring Manager.

*f)* Reallocate the bandwidth every interval by:

- First, call the reasoning procedure and get an action: *{$S:RellocateAction}* to disable or enable the low priority services.
- Second, share the bandwidth equally for every available service.

In b), c) and f), the Service Manager calls the reasoning procedure with an input XML expression based query Q. Using the CapOnt, SysCapOnt, inherent capabilities and require capabilities, the reasoning procedure transforms the query clause (2) to obtain the actions and the instantiations of the variables. Then, these actions and instantiations of the variables are returned to the Service Manager.

The Monitoring Manager gets the monitoring request, queries the parameters' values from the nodes, and updates these values to the Service Manager.

### *A. System-specific Capabilty Ontology (SysCapOnt)*

According to the standardized information models discussed in Section II, SNMP MIB and CIM schema have already a rich set of defined capability types and parameters. We choose the SNMP MIB as the basis for the definition of the capability types and the non-rule-based parameters in the SysCapOnt because of the extensive implementation of SNMP agents in different types of devices.

Note that this given ontology as illustrated in Fig. 4 is only for demonstration purpose. The capability types focused are network interface and operating system. These capability types and their parameters are found in Host Resource MIB [19] and MIB II [20].

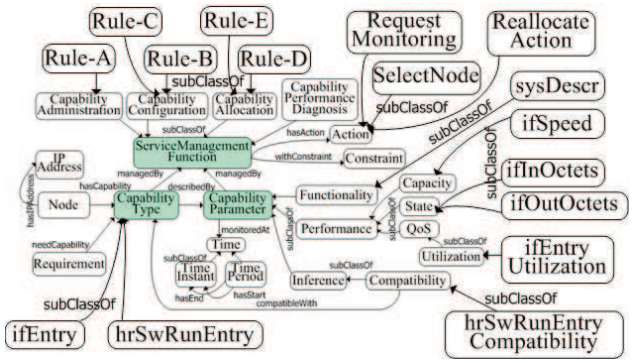


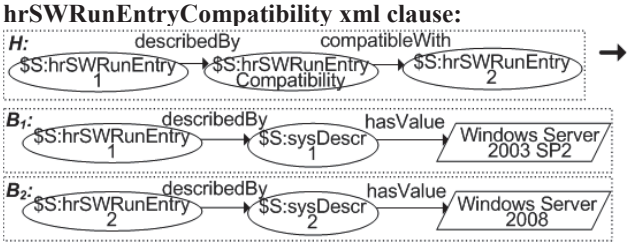

Figure 4. Some Concepts of a System-spefic Capability Ontology.

A network interface is defined by the object type *ifEntry*. It consists of several parameters, such as *ifInOctets*, *ifOutOctets* and *ifSpeed*. *ifInOctets* is the total number of octets received on the interface, whereas *ifOutOctets* refers to the total number of octets transmitted from the interface. *ifSpeed* is the maximum bandwidth of the interface. The

parameter *ifEntryUtilization*, a subclass of the Utilization, defines the bandwidth utilization of the interface. An operating system is defined by the object type *hrSWRunEntry*, and its type and version are described by the object type *sysDescr*. The parameter *hrSWRunEntry-Compatibility*, a subclass of the Compatibility, specifies the compatibility between operating systems.

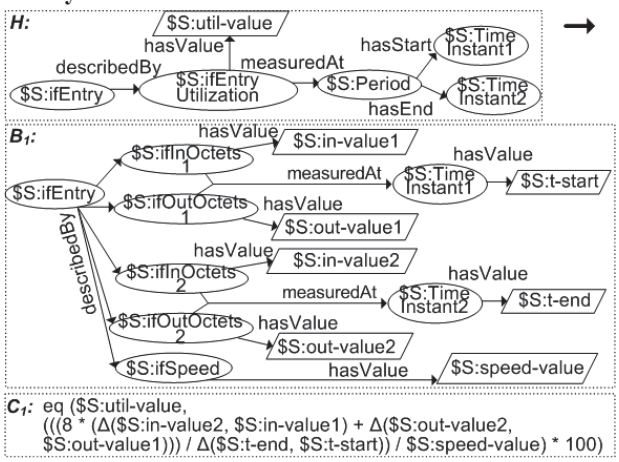
The service management functions are defined by *Rule-A – Rule-E*. These rules, the hrSWRunEntryCompatibility and the ifEntryUtilization are represented in the graphical XML clauses as below. The notations used are:


**hrSWRunEntryCompatibility xml clause:**




It can be read as: Windows Server 2003 SP2 is compatible with Windows Server 2008.

**ifEntryUtilization xml clause:**

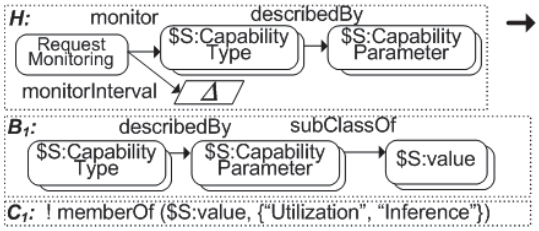



It can be read as: the bandwidth utilization *$S:util-value* during an interval, between *$S:t-start* and *$S:t-end*, depends on other parameters as this expression,

$$\$S:util\text{-}value = (((8 * (\Delta(\$S:in\text{-}value2,\$S:in\text{-}value1) + \Delta(\$S:out\text{-}value2,\$S:out\text{-}value1))) / \Delta(\$S:t\text{-}end,\$S:t\text{-}start)) / \$\$:speed\text{-}value) * 100 \quad (5)$$

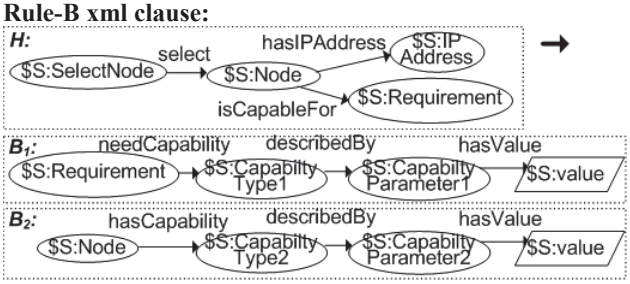
where *Δ($S:in-value2,$S:in-value1)*, *Δ($S:out-value2, $S:out-value1)* are the numbers of bytes received and transmitted via the network interface during an interval, whereas *$S:speed-value* is the maximum bandwidth.

**Rule-A xml clause:**



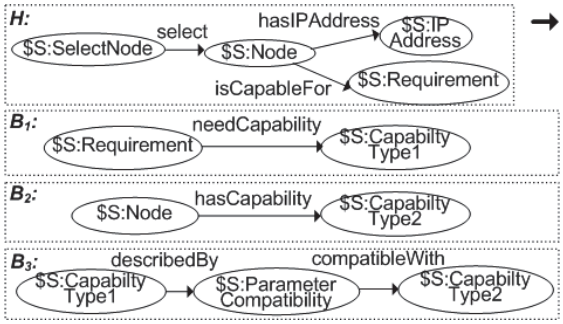

It can be read as: request for the monitoring of *{$S:CapabilityType, {$S:CapabilityParameter}}* every interval (Δ), if *$S:CapabilityParameter* is not the rule-based parameters: the Utilization- and the Inference parameter.

**Rule-B xml clause:**



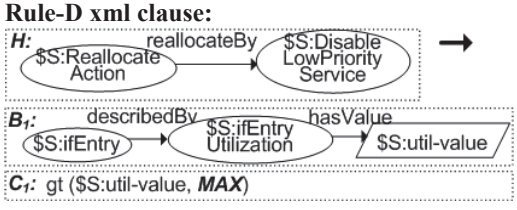

It can be read as: select a node *$:Node* as a capable node for the requirement *$S:Requirement* if it has capability types, parameters and values as same as required.

**Rule-C xml clause:**



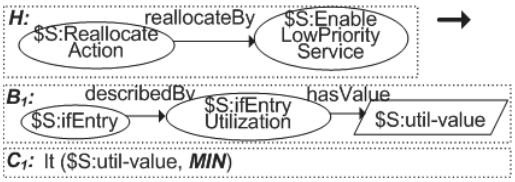

It can be read as: select a node *$:Node* as a capable node for the requirement *$S:Requirement* if it has capability types as compatible as required.

**Rule-D xml clause:**




It can be read as: reallocate the bandwidth by disabling the low priority services, if the bandwidth utilization is greater than a maximum limit (MAX).

**Rule-E xml clause:**




It can be read as: reallocate the bandwidth by enabling the low priority services, if the bandwidth utilization is less than a minimum limit (MIN).

### *B. Simulation Results*

We simulated our scenario by assuming that the nodes have the SNMP agents executing. The maximum wireless bandwidth is 54Mbps. The number of users is 100. Every user generates 2-5 service requests to transfer the videos *randomly*. All video files are the same size; 300Mb, and are transferred with the same maximum rate; 1Mbps. The interval (Δ) to monitor capabilities as well as to reallocate the bandwidth is 30 sec.

Our simulation is set for two cases (I, II). Both cases set low priority for the old videos as already mentioned. In case I, the system executes without the Rule-D and the Rule-E. The requests will wait if there is no bandwidth left. When there is released bandwidth, it is shared and given to all requests; processing requests and waiting requests, of the high priority videos before the low priority videos. In case II, the system executes with the Rule-D and the Rule-E. The old video transfer services will be disabled and enabled according to the bandwidth utilization. The maximum limit (MAX) and the minimum limit (MIN) are set as 80% and 60%. The average transfer times (Avr.T) in both cases are presented in Table II.

TABLE II. THE VARIABLE TYPES

| | **Avr.T of The Present Presentations** | **Avr.T of The Present Tutorials** | **Avr.T of The Old Videos** |
|---|---|---|---|
| Case I | 6.82 min | 6.26 min | 9.73 min |
| Case II | 5.76 min | 5.39 min | 14.95 min |

In Case II, the system degrades the services more efficiently than in Case I. The system can offer an adequate transfer of the high priority videos. The average transfer times are 5.76 min and 5.39 min, which are faster than in Case I ≈15.5% and ≈13.9% respectively. These %values indeed vary according to the arrival distribution, Δ, MAX and MIN. However, the average transfer time of the low priority videos in Case II (14.95 min) is greater than in Case I (9.73 min). This is because in Case II the reasoning procedure executes the rules and suggests the action to pause their transfers when the bandwidth utilization > 80%. The bandwidth that has been used is released, and is given for transferring the high priority videos. But in Case I the bandwidth is released only when the transfers finish.

## VII. CONCLUSIONS

Capability Ontology (CapOnt) comprising the concept capability types, capability parameters and service management functions is proposed. The rule-based parameters values, e.g. the Utilization and the Compatibility, are generated dynamically by the presented reasoning mechanism. The capability types and the non-rule-based capability parameters can be defined based on the existing standardized information models. An intelligent conference room example, where the Capability Ontology combined with the reasoning mechanism has proved useful, is presented. A System-specific Capability Ontology (SysCapOnt) for a conference service system is also given. The conference service system comprises the adaptation in case the bandwidth utilization is greater than or lower than the limits. Using the proposed rule-based service management functions, the system (re)-allocates the bandwidth to the high priority service classes more efficiently.

## REFERENCES


[1] F. A. Aagesen, P. Supadulchai, C. Anutariya, and M. M. Shiaa, "Configuration Management for an Adaptable Service System," In Proc of IFIP Int. conf. on Metropolitan Area Networks, Architecture, Protocols, Control and Management, Vietnam, April, 2005.

[2] K. McCloghrie, D. Perkins, and J. Schoenwaelder, "Structure of Management Information Version 2," RFC 2578, 1999. Available at: http://www.ietf.org/rfc/rfc2578.txt [Last accessed June 2010].

[3] DMTF, "Common Information Model (CIM) Standards," Available at: http://www.dmtf.org/standards/cim/ [Last accessed June 2010].

[4] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge Engineering: princicples and methods," in Data & Knowledge Engineering, vol. 25, pp. 161-197, 1998.

[5] P. Supadulchai, "NxET Reasoning Engine," Plug-and-play Technical Report, Department of Telematics, NTNU, ISSN 1500-3868.

[6] W. Stallings, SNMP, SNMPv2, SNMPv3, RMON 1 and 2. The 3rd edition, Addison-Wesley, 1999.

[7] C. Elliott, D. Harrington, J. Jason, J. Schoenwaelder, F. Strauss, and W. Weiss, "SMIng Objectives," RFC 3216, 2001.

[8] ITU-T Recommendation X.722, Information technology - Open Systems Interconnection, "Structure of management information: Guidelines for the definition of managed objects," January 1992.

[9] J. E. López de Vergara, V. A. Villagrá, and J. Berrocal, "On the Formalization of the Common Information Model Metaschema," In Proc of the 16th IFIP/IEEE Int. Workshop on Distributed Systems: Operations and Management (DSOM'05). LNCS, vol. 3775, pp. 1-11, 2005.

[10] J. E. López de Vergara, V. A. Villagrá, and J. Berrocal, "Applying the web ontology language to management information definitions," In IEEE Commun. Mag., vol. 42, no. 7, pp. 68–74, Jul. 2004.

[11] J. Keeney, D. Lewis, D. O'Sullivan, A. Roelens, A. Boran, and R. Richardson, "Runtime semantic interoperability for gathering ontology-based network context," In Proc of IEEE/IFIP Network Operations and Management Symposium (NOMS'06), pp. 56-66, Canada, 2006.

[12] A. K. Y. Wong, P. Ray, N. Parameswaran, and J. Strassner, "Ontology mapping for the interoperability problem in network management," In IEEE Journal Sel. Areas Commun. 23(10), pp. 2058-2068, 2005.

[13] A. Guerrero, V. A. Villagrá and J. E. López de Vergara, "Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL," In Proc of the 16th IFIP/IEEE

Int. Workshop on Distributed Systems: Operations and Management (DSOM'05). LNCS, vol. 3775, pp. 12-23, 2005.

[14] H. Xu and D. Xiao, "A Common Ontology-Based Intelligent Configuration Management Model for IP Network Devices," In Proc of the 1st Int. Conf. on Innovative Computing, Information and Control (ICICIC'06), China, 2006.

[15] I. Diaz, C. Popi, O. Festor, J. Tourino, and R. Doallo, "Ontological Configuration Management for Wireless Mesh Routers," In Proc of the 9th IEEE Int. Workshop on IP Operations and Management (IPOM'09). LNCS, Vol. 5843, pp. 116-129, 2009.

[16] W3C, "OWL Web Ontology Language Overview," 2004 Available at: http://www.w3.org/TR/owl-features/ [Last accessed June 2010].

[17] P. Supadulchai and F. A. Aagesen, "Policy-based Adaptable Service Systems Architecture," In Proc of the 21st IEEE Int. Conf. on Advanced Information Networking and Applications (AINA'07), Canada, 2007.

[18] V. Wuwonse and M. Yoshikawa, "Towards a language for metadata schemas for interoperability," In Proc of the 4th Int. Conf. on Dublin Core and Metadata Applications, China, 2004.

[19] S. Waldbusser and P. Grillo, "Host Resource MIB," RFC 2790, 2000, Available at: http://www.ietf.org/rfc/rfc2790.txt [Last accessed June 2010].

[20] K. McCloghrie and M. Rose, "MIB II," RFC 1213, 1991, Available at: http://www.ietf.org/rfc/rfc1213.txt [Last accessed June 2010].