Defence Research and Development Canada   Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Evaluation of Two Experimental Policy Based Network Management Systems

K. Bodnar

Defence R&D Canada – Ottawa

CONTRACT REPORT

DRDC Ottawa CR 2005-108

July 2005

Canada

Evaluation of Two Experimental Policy Based Network Management Systems

K. Bodnar
NRNS Incorporated

NRNS Incorporated
4043 Carling Avenue
Ottawa, ON
K2K 2A3

Contract Number: W7714-3-800/001/SV

Contract Scientific Authority: Ms. Isabelle Labbé, CRC (613) 990-1410

Contract Scientific Advisor: Dr. S. Zeber, DRDC Ottawa (613) 991-1388

The work described in this report was sponsored jointly by the Department of National Defence under the work unit 15BF and by the Communications Research Center Canada under the work units 15CS and 15CV

Defence R&D Canada – Ottawa

Contract Report

DRDC Ottawa CR 2005-108

July 2005

Document Revision History

Version 0.1	25-October-2004	Partial DRAFT submitted to Technical Authority for review.
Version 0.2	09-November-2004	Full DRAFT submitted to Technical Authority for review.
Version 1.0	20-November-2004	Final submitted to Technical Authority.
Version 1.1	17-July-2005	Added additional comments from DRDC Ottawa.

Table of Contents

1. Introduction

This report is prepared for Communications Research Centre (CRC). It is an evaluation of two experimental Policy Based Network Management (PBNM) Systems. The goal is to evaluate the systems and to assess their suitability for use in the development of a solution for a generic PBNM system.

1.1 Background

The Communications Research Centre (CRC) has started a research effort on a Policy-Based Quality of Service (QoS) Management System for mobile networks. The system investigates the deployment of high-level policies that promote dynamic end-to-end Internet Protocol (IP) QoS and advanced path selection in a deployed maritime tactical network.

Defence R&D Canada (DRDC) Ottawa and CRC have initiated a cooperative activity to explore the possibility of developing a generic PBNM system architecture that would provide an integrated support for security and QoS policies.

There are two experimental systems under consideration. The first is known by the acronym CNMS or Coalition Network Management System. This system was developed by the Australians and integrates a series of COTS (Commercial, Off the Shelf Software) along with some modules written in Java, including the graphical user interface (GUI).

The second system is known by the acronym UMU-PBNMS and is a Policy Based Network Management System under development by the University of Murcia in Spain. This system is a 100 % Java implementation with very little COTS inclusions.

1.2 Purpose

This report presents the findings of an investigation to determine the extensibility of each system with the intent of creating a full-feature generic Policy Based Network Management System. There will be a high level comparison of architectural features as well as a general assessment of Reliability, Vulnerability and Maintainability of the two development models.

1.3 Caveats

The recommendations presented in this report are based on a time-limited investigation of 15 days per tool. The CNMS system was not totally run ‘in situ’ because the lack of a J-Rules runtime in the lab environment where the evaluation was taking place. The UMU – PBNMS was a crippled demo. While communication was demonstrated between the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP), no real configuration or policy work could take place because the demo was tightly tied to the test bed at the University of Murcia.

1.4 Scope

The original scope was to consider only the Java extensibility of both applications. However the scope was expanded to include a high level comparative study with some low level architectural details. Reasons for this scope creep was that in one system the Java code was used primarily as an integration and GUI method, while in the second instance, the entire application was Java based.

2. Coalition Network Management System (CNMS)

2.1 CNMS Description

The core of this application is described in TTCP-C31-TP8-2004-PA-3. The demo and experimental application consists of a Domain Policy Integration Manager that defines a Network Management Policy environment. This environment consists of a domain under management with various control elements such as a Policy Decision Point, a Policy Enforcement point, Management Control Interfaces (MCI), Element Integration Modules and a variety of virtual and functional devices running over a topology of networked nodes.

Essentially, a high level policy is accessed from a secure repository and entered into the Domain Policy Integration Manager (DPIM). The heart of the architecture is a series of transactions whereby the policy decision and distribution point approves the policy synthesis or translation from a high level policy to a low level policy and transfers that policy to a Policy Enforcement Point. Policy configurables such as ancillary services and applications like Bandwidth Broker and Low Bandwidth Manager can also be managed.

The functionality described above requires an extensive list of infrastructure services, applications and protocols. Application components consist of HP OpenView (HPOV) as a COTS network management tool capable of implementing low level policy parameters in the network. Tied to that, a Naming Service is required, implemented with OpenOrb. OpenOrb also implements the Event Factory Service. OpenOrb is run from a Java jar and the JDK1.4.x SDK. A Shared Data Model (SDM) Common Object Request Broker Architecture (CORBA) server acts as an interdomain messaging proxy for the DPIM (and incidentally the Bandwidth Broker). The actual Policy Enforcement Element Integration Manager (EIM), in this case is simply a CORBA wrapper for HP OpenView.

In terms of functional detail of policy formulation, synthesis and enforcement, the policies, both high level and low level are stored as eXtensible Markup Language (XML) documents. An ILOG JRULES engine parses the high level document and outputs a low level XML policy based on an encoded semantic rules-parsing fed into the engine as a series of parameters. It saves code recompiling because it serializes and deserializes the rules parameters. So while the logic of what any given software tool does not change much after conceptual design -- the parameters do. The parameters are serialized or marshalled into XML and read in at runtime and deserialized. Thus if you change the parameters, you don't have to recompile the whole code.

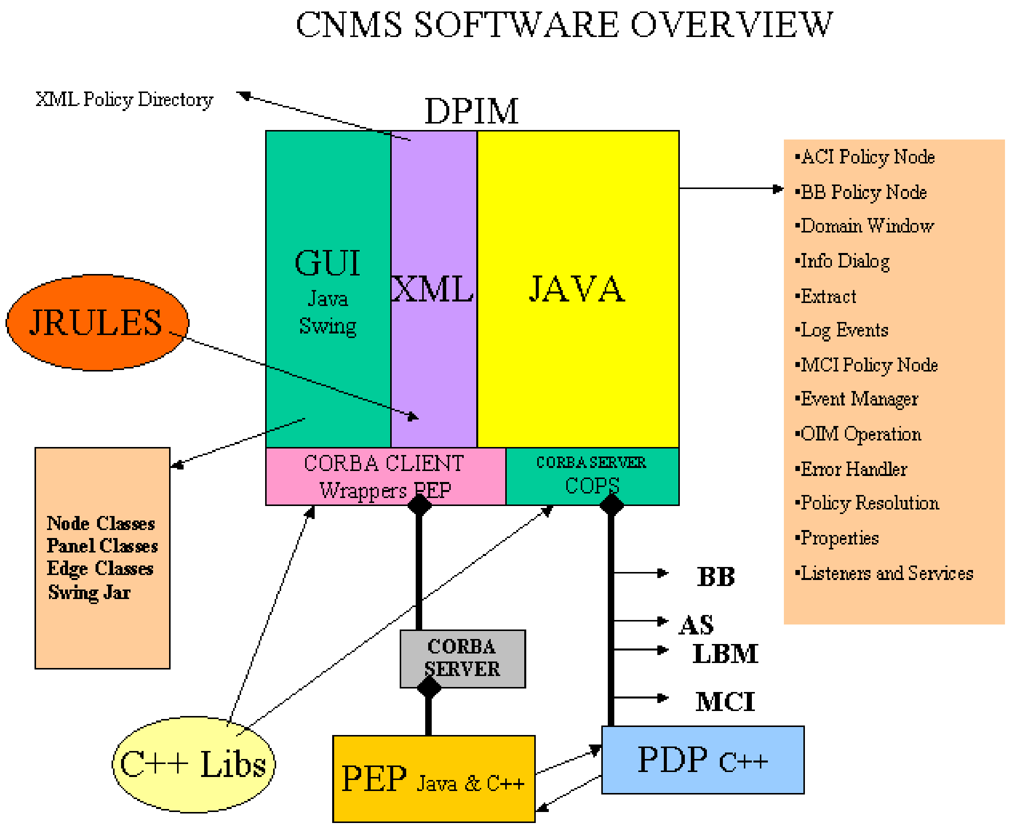
2.2 CNMS Dependencies

While this list is relatively complete, it by no means can be considered canonical, but it is sufficient for the scope of this study, as these were the dependencies for the demonstration.

- JDK 1.4.x (DPIM & PEP)
- JRE 1.4.x (DPIM & PEP)

- Boost C++ Libraries - Specifically the boost::thread library (COPS implementation)
- HP Open View (PEP)
- Solaris or Linux (OS Platforms)
- Win,NT,x86 (OS Platforms)
- ACE ORB (TAO), CORBA V2.6 compliant, C++ ORB (Object Request Broker)
- OpenSSL – for security, certificates
- Breeze – Java components, including XML Binder for Java
- Perl -various scripts to make the Open SSL work
- Apache Avalon Framework
- Avalon Excalibur Classes
- ANT - ANT version 1.5.3.
- OpenORB - OpenOrb 1.3.1
- ILOG JRULES rules engine

2.3 High Level Diagramatic Overview of CNMS



2.4 CNMS Findings

This application demonstrates an amorphous accretion of open source, COTS and proprietary code parts integrated by Java J2EE classes. The integration of all of these parts required an obvious virtuoso display of technical proficiency and prowess in multi-disciplinary aspects of design and development of network components and protocols. From the integration of Common Open Policy Service (COPS), to CORBA wrappering of HP OpenView, to a JRULES rules engine, it is obvious that the system was cobbled together by subject matter experts.

There are subtle functionalities that are not visible at first blush that give this application a depth and richness of features. Examples of this include Corba Client inter-domain signal monitoring, Name Server Browsing, and object wrappering for security through MCI transmission points.

That having been said, there are several weak spots in algorithm implementation. For example, XML binding was used by a proprietary Java jar that eliminates DOM expansion. In plain English, this represents a potential risk when the system has to be maintained or upgraded by person not familiar with its development.

There is a multitude of development environments and operating systems covering the gamut from Win to NT to Linux to UNIX. It was a surprise to see the Apache Avalon Framework hidden in the classes. This is an open source Java Framework embodying

Component Based Design which is supposed to be the next step to Object Oriented Programming. The framework gives the objects the ability to manage themselves, so to speak, by containerization with custom management classes. In addition to this was the Apache Excalibur jar that accompanies the Avalon framework adding classes that provide threading and other functionality. Then there is the ANT make tool for the Java classes.

Many DLLs written in C++ are required for this application. The installer registers a whole directory of them. The DLLs are Boost open source libraries that offer off-the-shelf functionality. There are the TAO Corba C++ libraries. The DLLs fell into the following categories (**bolded are the functional dependencies**):

- Bin **(Messaging, Events on DPIM)**
- Boost **(Boost Libraries for COPS implementation)**
- Tao **(Corba Implementation)**
- Ssl **(Open SSL for checking out Policies from the Repository)**
- Ace **(DPIM and PDP)**
- server.config **(infrastructural DPIM)**
- libeay32.dll **(Win support)**
- ssleay32.dll **(Custom for OpenSSL)**
- runtime/debug dll's **(C++ Libs)**

The biggest workhorse of the COTS lineup is HP OpenView, the network management tool that is responsible for policy implementation and enforcement. HPOV also has its own messaging protocols in monitoring and managing the networks. **Currently in the demo, HP OpenView is the sole Implementation of a Policy Enforcement Point. While the architecture defines OpenView as simply another PEP, in the practical implementation of the demo, OpenView is the only means of enforcement and implementation offered. It is so tightly integrated into the tool, such that the DPIM has hard-coded CORBA wrappers of OpenView commands.**

The custom java code itself was used to provide a GUI and as an integration tool. It could hardly be described as the foundation language of the tool. Many of the carrier protocols, message protocols, signal protocols and various bits and pieces were provided by open source or proprietary tools.

The GUI itself is novel, graphical, intuitive and replete with features that are impressive, such as graphical entities changing colours, stretching when relocating with a mouse etc. However, there was a wide variety of customizable components, but there was no way of remembering them, as in a preference file. Also some of the graphics were not kerned or cut properly and there is annoying pixilation around the edges.

It was more than a little surprising to see how little attention was devoted to using a browser as an adjunct tool in this application. Because it isn’t monolithic, and the main

Policy Management Tool is thick client, web-centric methodologies would not lend themselves well to the melange of technology.

Perhaps the demos were crippled, but there was no user authentication. Roles could be defined but I didn't see any evidence of pre-defined roles that would be a jumpstart to plug and play.

Policy Storage consisted of entries in a directory. In a diverse element, diverse domain topology, a database would be logic solution to store various high and low level policies, along with their descriptions, parameters and usage metrics that would increase the useability of the system.

As for security, the policies come from a secure repository, but the demo did not check certificates and authorizations at various steps in the PDP to PEP chain. Open SSL, an open source tool that implements Secure Socket Layer is used for secure session establishment and management.

The program is multi-threaded which is considered desirable since there is a lot going on in terms of transaction and module to module communication.

There are a series of static text files which require configuration parameters to be entered. The location and content of these configuration files is not intuitive.

Several utilities are provided such as EIM, which is a stand alone Corba client for inter-domain signalling, a PDP client policy loader, runnable test PEPs, a trap sender to a HP OpenView machine, and wadjet – a key tool to kludge certificates.

2.4.1 Java Code Findings

The source code of all of the Java classes was examined. The Java occurs in two places, the PEP (Policy Enforcement Point) and in the DPIM (Domain Policy Integration Manager). It should be noted that the PEP has both a Java Component and and a C++ component. The PEP java classes are the PEP server, the listeners, the shutdown hooks, and the inflaters and deflaters. DPIM java elements in the PEP include the MCI interface and the C Object interface as well as the COPS PEP interface and policy event and listener java classes.

The DPIM java classes and source code consists some 30 classes for methods and objects such as Policy Holder, Policy Resolution, Policy Print, Shutdown hooks, Utils, XML content, the main GUI which is an extension of the JPANEL class, operator interface module (OIM) operations, Object Properties Frame and a series of messenger helpers, listeners and communicators. In addition there are extensions of Swing components, Image Utils and XML helpers in the source code set. In all there are 60 Java classes of various complexities.

Since the central requirement of the study was to evaluate extensibility and conversion and refinement to a generic policy based network management system the Java code was evaluated as to how well it was coded, it's structures, and its adherence to good coding practice. Below are the findings:

- Where errors are indeed trapped, there are no methods or reporting of the errors.

- Similarly there is no drill-down for internal event messages when the 'try' method fails and exceptions are caught.
- There is very little comments and very poor javadoc comments for documentation.
- There is no documentation provided with the code.
- Coding templates were not used.
- Very little evidence of version control in the headers and configuration control.
- Coding was done to classes and not to interfaces. This doesn't promote flexibility or extensibility.
- Profusion of use of unchecked exceptions.
- On the plus side, objects and methods have clear responsibilities and are well compartmentalized.
- Little logging or de-bug methods.
- Little or no code duplication.
- More of the Interface Definition Languages (IDLs) could have been effected in Java rather than in C++.
- The source code files were not derived from templates, making for a diversity in morphological structure of the code.

2.5 CNMS R.A.M. & Vulnerability

2.5.1 Reliability, Availability and Maintainability of Code

The large number of dependencies and the existence of proprietary software makes this system rate low in Reliability, Availability and Maintainability (RAM). Installation of the various COTS components from scratch would be an onerous affair. It would have to consist of installation, configuration and then integration with the rest of the application.

There is a significant effort in re-compiling for all of the different platforms should an Operating System be upgraded. One would also run the risk of deprecated API's both in the open source and in the proprietary software such as JRULES. The JRULES licence is quite expensive, making this a high cost solution.

In the event of an open source upgrade, considerable time may be spent in integrating a new component among the older other elements of the application.

In any field deployment, the complexity factor of all of the integrated COTS would make debugging the system a nightmare, especially if in the dire consequences of a tactical deployment if the system malfunctioned and had to be fixed in a hurry.

2.5.1.1 Reliability

There are too many individual points of failure ranging from the JRULES engine, to the custom PEP server, to the Open Orb or the Corba Server, or an HP OpenView malfunction. The Mean Time Between Failure when factored with all of the components could be quite low. Thus this application is rated low in reliability.

2.5.1.2 Availability

Because of the possibility of diverse Operating Systems failing (for example the UNIX server running OpenView fails) then the availability of the entire system is affected. The diversity of elements and the lack of parallel redundancy in the architecture would make calculated availability parameters quite low on a system like this. Thus this application is rated low in availability.

2.5.1.3 Maintainability

If the various component Operating Systems suddenly needed security patches that affect the operation of the PBNMS, maintainability in the field could be a nightmare. If an OS patch is applied and the system stops behaving properly, there could be a conflict of requirements between the OS and PBNMS. It would be an extremely difficult problem to solve at the deployment level. Since there are so many unknowns, this application is rated low in maintainability.

2.5.2 Vulnerability

Because of all of the COTS and protocols, if one were to do a port scan on application and systems, the results would give one pause to consider. There are ports for COPS, HP OpenView has several dedicated ports, the network protocols have their ports, and the event listeners and domain signallers all jump into the chorus.

Obviously a system like this would need its security provided elsewhere, such as by firewalls, routers, demilitarized zones (DMZs) and bastion servers guarding perimeters. HP OpenView itself is not a particularly secure EIM in the scheme of this application.

s it sits, because of the COTS and accretion of the libraries and applications, a full list of open ports and vulnerabilities is not immediately known unless a full blown Threat and Risk Assessment is done against the entire system with automated tools.

A more monolithic system under development from a central authority would have a better handle on mitigating the vulnerability risks.

2.6 CNMS Known Bugs

1. When loading a High Level policy such as the example provided, 'HLPolicy Data Example.xml', an exception is throw with the message: 'Load HL Policy Error .... Domain: CAN not found'. For a first time load, it doesn’t think that it has relevancy to the current domain. This is a known bug.

2. When using the PDP with the java_pep example, one may get authentication errors even though both sides don't require authentication. One may need to set up OpenSSL and sign the javapep's client certificate with the CA created in the ssl directory and then point the OpenSSL environment variables at the CA. However this is the developer's theory and he states that he didn't have time to test it.

2.7 CNMS Summary of Findings

The java code is not very extensible in its present form. All of the classes would have to be re-visited to add error trapping, comments and interfaces rather than calling direct classes. Also reflection should be implemented rather than Factory methods which will aid in portability and cross platform effectiveness in the byte code.

There is too much of a mix of technologies here to make the CNMS system extensible. One could keep adding on functionality, but each addition would be an appendage rather than a component module of a planned whole. In the author's opinion, there are too many points of failure. The demo which is a miniscule subset of functionality took a couple of days to install. The full blown system installation would require a substantial level of effort and considerable expertise. This is definitely not plug and play.

2.7.1 Modularity and Extensibility of the CNMS System

While the application consists of a series of tightly integrated modules, the modularity of further design implementation – i.e. tinkering with the internals is at best, a difficult proposition in its present state. Adding different instances of existing functionality doesn't seem to pose a problem if the configurations are set properly in the files, but a new element introduction may be an entirely different matter.

For example, the CORBA wrappering of the PEP HPOV is quite specific. The inter-relationship between the DLLs has a specificity that defies an empirical account until one actually tries to add a new module. The interfaces are poorly defined, and documentation, especially on open source components is lacking.

From purely a project risk point of view, the extensibility of the present incarnation of the demo of the CNMS is deemed to be of a low order.

3. University of Murcia – Policy Based Network Management System (UMU-PBNM)

3.1 UMU-PBNM Description

The UMU-PBNM system is a research project of a monolithic system built with Java with very little COTS. The COTS elements consist of entirely of Java-based elements such as a JAIN-SIP Session Initiation Protocol (SIP) based signalling protocol and IAIK Secure Sockets Layer (SSL) developed in Austria.

Essentially the policy management system is an integrated thin client, web-centric tool consisting of the usual elements of a policy repository, a policy creation algorithm, a policy decision point and a Policy Enforcement Point. In both systems, policies are stored as XML documents and passed around the topology as such until they are decoded into PEP commands.

The Graphics User Interface is Hyper-Text Markup Language (HTML) with Java Server Pages (JSP). Rather than an algorithm from a high level to low policy translation, the current incarnation has a JSP forms HTML policy editor that takes the textual input and converts it to an XML policy using string manipulation. Currently the policy generation capability is a low-level policy of parameters, almost at the device level.

The system authenticates from a PKI server using an SSL algorithm. This authentication happens at almost every step and click of the mouse on the web page. It has a definite policy structure schema based on roles and security levels attached to those roles. For example in the demo, a pre-defined user role could be assigned to a LOW, MEDIUM or HIGH security level that helps define the policy and its implementation.

The UMU-PBNM is not as specific to a tactical or military environment as the CNMS is. It is a much more abstract implementation of a PBNMS.

The demonstration system consists of the policy management tool, the PDP server and a PEP server that runs in Java consoles. The demo of the PDP to PEP is merely a messaging demonstration to prove their successful implementation of COPS called JCOPS, which was developed at UMU. The PEP on the UMU test bed consists entirely of a shell script of Linux based commands.

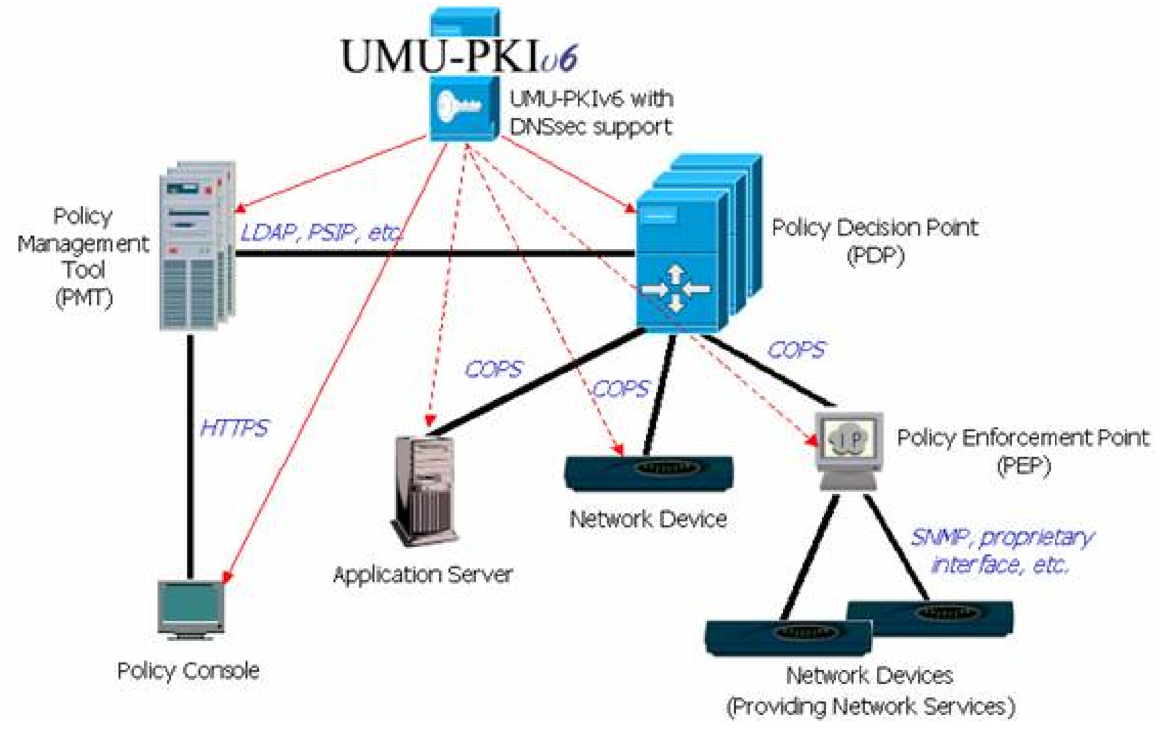
The effort of the development of this system went not to the features but rather the building of the required infrastructure from first principles Java, such as the JCOPS implementation. Hence things like QoS are just currently motherhood statements – a gleam in the eye of the developer.

From an architectural perspective of what one could see, it was a less complicated system than the CNMS system. This is from a high level. The java implementation of many required infrastructure items, such as the COPS, is more technologically elegant with a much smaller footprint.

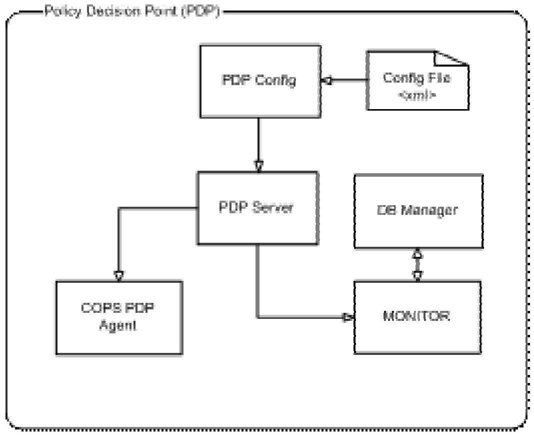
However the lack of implemented features precludes comment on how well the system functions as a unit. As of yet, functions such as policy synchronization, inter-domain communications, policy resolution and primitive policy conflict checking are not implemented in the PM tool, however the capability can be made present.

The demo was barely a demo and tightly integrated to the UMU test bed. This is not a reflection of its extensibility, but rather the demo was the first one ever disseminated to an outside party by the university, and it was hastily cobbled together, and the packages were changed in mid-stream.

3.2 Overview Graphic of the UMU PBNMS (as supplied by UMU)

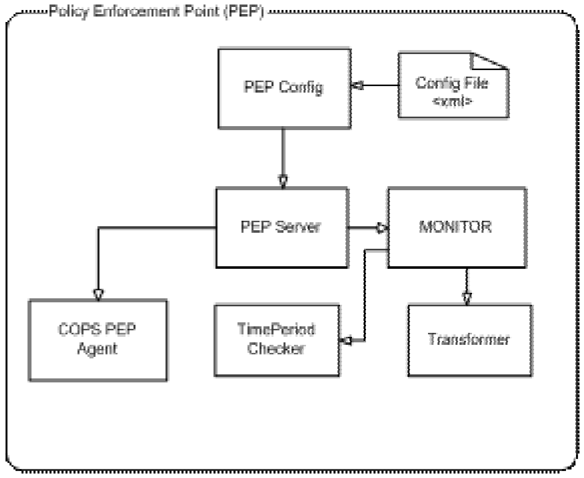

UMU-PKIv6
UMU-PKIv6 with DNSsec support
Policy Management Tool (PMT)
LDAP, PSIP, etc
Policy Decision Point (PDP)
COPS
COPS
COPS
HTTPS
Policy Enforcement Point (PEP)
Application Server
Network Device
SNMP, proprietary interface, etc.
Policy Console
Network Devices (Providing Network Services)

3.3 UMU PBNMS PDP Schematic



Fair comment on this is limited as the demo never ran properly. It was tightly tied to the UMU testbed. This code is implemented in primitive form, however it would require additions to make it fully functional.

3.4 UMU PBNMS PEP Schematic



Same as above. Fair comment on this is limited as the demo never ran properly. It was tightly tied to the UMU testbed. Again, this code is implemented in primitive form, however it would require additions to make it fully functional.

3.5 UMU PBNM Dependencies

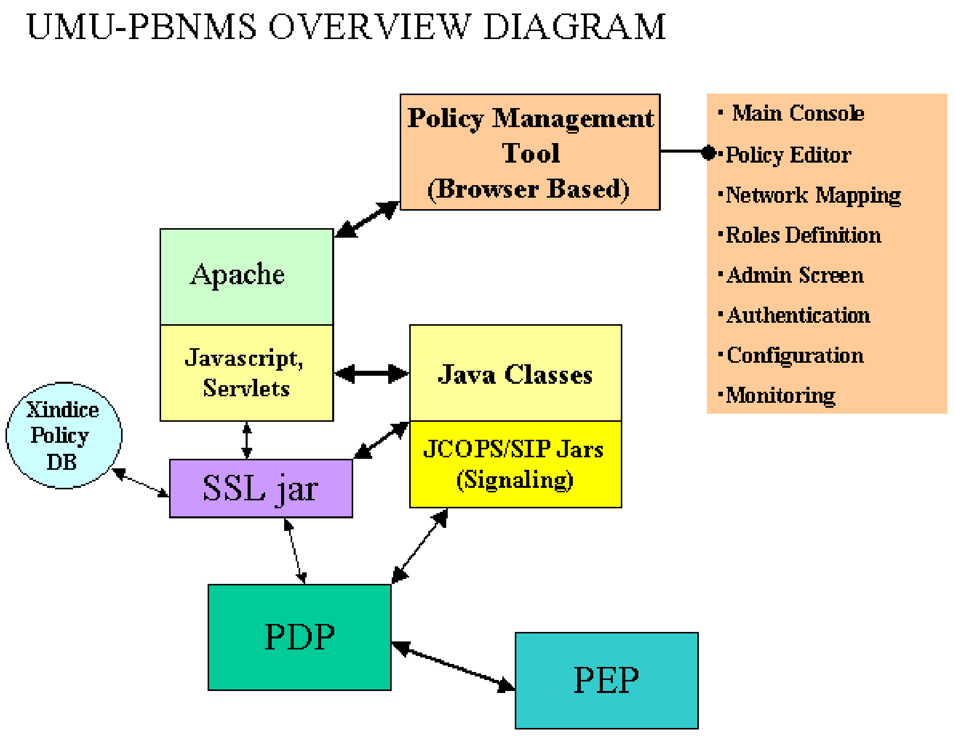
The dependencies of the UMU-PBNM were all java based. The list is as follows:

- commons-modeler Jars
- Realm Jars
- iaik jar

- mx4j jar
- servlets
- tomcat coyote
- tomcat util
- tomcat warp
- ant
- infozone tools
- xalan
- xerces
- xindices
- xml apis
- xmldb

Of these, the commercial jars are the mx4j jar used in topology self-discovery, the JAIN-SIP jar used in Signalling Interface Protocol, the infozone tools for network management and the IAIK SSL implementations. Most of these commercial implementations have open source counterparts.

3.6 High Level Overview of the UMU-PBNMS System



3.7 UMU PBNMS R.A.M. & Vulnerability

3.7.1 Reliability, Availability and Maintainability of Code

The dependencies of the UMU – PBNMS are few and are native to the code. The net effect is that the pieces of the heterogenous system do not have a large degree of complex interactions of diverse technology. Thus the application is fairly straight forward in its Reliability, Availability and Maintainability.

The installation is straight forward and requires the servlets to be put into the Apache home, and the Java jars to be included in the classpath. Other than that there is no complexity to the installation. Deprecated API's in Java run gracefully with any JRE runtime. There is little here to go wrong once the system is tested and deployed.

3.7.1.1 Reliability

There are very few individual points of failure in the UMU – PBNMS system.. The Mean Time Between Failure because it is monolithic Java byte code, is very high. Thus this application is rated excellent in reliability.

3.7.1.2 Availability

Since all of the services are Java and Java related the availability of the application is high. The availability subsumes the availability of the Apache server which serves up the thin client. The Apache server is the most vulnerable component in terms of availability, but this reflects well on the application.

3.7.1.3 Maintainability

The maintainability of this application is rated as excellent. There are very few discrete components to maintain, and those that do exist can be maintained by replacing the jar that provides their services.

3.7.2 Vulnerability

The Graphics User Interface of this application is a thin client and browser based. Hence the most vulnerable aspect of this application is not within the bounds of the application itself, but rather within the Apache server that delivers it to the users browser. However if the service is sufficiently hardened, then the vulnerability is rated quite low.

3.8 UMU PBNM Findings

As previously stated, the demo was crippled and specific to the UMU testbed. The code itself vacillated between two poles. On one hand there was sublimely written, technologically elegant classes, all written from source code templates that were highly organized and easily readable. On the other hand, there were obvious kludges like the hard-coding and compilation of host names into classes.

The specificity to the UMU testbed is not a feature of the system but rather a feature of the demo implementation.

There were 32 current JSP modules, and 117 java classes coupled to 14 jars required for the current incarnation of this tool. In terms of magnitude, the footprint of the application is quite small, in relation both to the CNMS and the infrastructure functionalities required for a PBNMS. The XML policies were stored in a native Xindice database which is a sensible and logical method, rather than keeping a rag-tag policy collection in a directory. It is searchable with intelligent XML Path (XPATH) commands that have the capability of embedded functions.

The features were much more primitive than the CNMS, and while the intrinsic embedded SSL security was tighter than CNMS, there is room for improvement of interfacing to boundary devices such as the CNMS MCI.

3.9 UMU PBNM Summary of Findings

The UMU effort was based on developing the infrastructure rather than the features. As a result, the system is highly extensible. From an algorithms point of view, because of the design, virtually any sort of Java module could be added. But in reality, very few features are implemented.

The groundwork for the implementation of features is laid. There are no major missing parts (such as event factories or underlying technologies) that would preclude the system from functioning once the features were implemented. The JCOPS implementation was developed at the University of Murcia, and an open source GNU Public Licence (GPL) may be available for further development. Also Remote Method Invocation (RMI) is used instead of CORBA. RMI allows the full suite of registration services (such as PEPS registering to the PDP). A service implemented as an RMI object allows a remote object to register its stub, and received data without wrappering. Because it is Java to Java, it has a smaller footprint and it more versatile a method than wrappering for CORBA.

What code there is, is well written and quite extensible. It is commented (in Spanish) and has a better set of Javadoc components than the Australian Java.

While there would be a serious level of effort required to bring the demo to the same level as the CNMS, there would be much less effort required in installing, configuring and running it.

The UMU-PBNMS has its origins in theoretical underpinnings. As a result, few requirements are currently satisfied by the corporeal body of code. However it does have all of the hooks and ports to do so.

4. Comparative Study of Architectural Features between CNMS and UMU PBNMS

Feature	***CNMS***	***UMU PBNMS***	***Comments***
Graphical User Interface	Thick Client	Thin Client, n-Tier Web Centric	The graphical pictorial interface of the CNMS is well suited to the tactical environment. It is quite intuitive and is impressive.
Approach	COTS, multi-part technology	Monolithic Java	The COTS methodology gives much more features to the demo but detracts from extensibility
User Authentication	None in demo	Certificate required for almost every function in admin mode.	Tighter SSL security in UMU PBNMS
Roles	Infinite, user-defined, can be manually input, co-relates to low level policy.	Pre-defined roles each associated with low level policy. May be edited.	Pre-defined roles part of the architecture
Protocol	COPS	JCOPS implementation	Elegant JCOPS implementation – not complete, but extensible
Policy Storage	Directory	X-Indice Native XML Database	The native XML database is a stroke of architectural genius
Transactions	CORBA, open ORB	Java RMI	Simpler implementation with less compute dynamics required for Java RMI
Policy Formulation	High Level to Low Level Policy Translation using JRULES rules engine and XML parser	JSP HTML Input form for the synthesis of low level policy document	The policy formation is quite embryonic in the UMU incarnation
Topology	Dynamic Capable, Require Topology File Configuration	Currently Static	The UMU has the promise of built-in topology self discovery, but not implemented.

Security	OpenSSL COTS	Embedded function COTS	The certificates are referenced at every step in the UMU system (possibly because either the design is stateless or a state server has yet to be implemented). However it is a more secure implementation.
Signalling & Sessions	SDM – Signalling Dispatch Manager	JAIN SIP Java Protocol	Small footprint of the JAIN SIP protocols
Application Topology	Multi-threaded implemented in C++	Non-threaded currently	The UMU incarnation is not yet multi-threaded
Vulnerability	Multiple Management Ports – COPS, HP Openview, SDM ports	Fewer ports – JCOPS (same as COPS ports) and no dedicated ports for JAIN SIP	The UMU incarnation has less chance for vulnerability because it is less diverse.
Java Development	Avalon Framework, Avalon Excalibur Classes	First Principles J2EE	The development of monolithic java is preferred to the multiplicities of the COTS infrastructure.

5. Overall Summary

If the overall summary required one sentence, it would be this: "The CNMS is a feature-rich system with a cobbled-together ad-hoc infrastructure, while the UMU-PBNMS has a solid, built-from-scratch infrastructure with few features."

Each system would require extensive development, or re-development to be properly deployed in a tactical environment. It is felt that ultimately the extensibility of the UMU-PBMNS would be more advantageous, easier to deploy and maintain and retaining the ability of easy upgradeability as the requirements scope changes.

If one were to pick a system for further investigation and development, the UMU – PBNMS is the recommended choice because of its superior inherent extensibility.

Annex A – Architectural Details for a Synthesis of an Ideal Generic PBNM System

If one were to cherry-pick the ideal features gleaned from the examination of both systems, the following would be incorporated into the generic PBNMS.

1. Thin Client for easier deployment to hailing networks.

2. Native XML database for policies – Xindice

3. Both a policy derivation/ extraction / translation system via rules AND an integrated policy editor both high level and low level.

4. Security and authentication at each step.

5. Entire development in Java

6. A library of roles, with a searchable X-PATH database. For example, if there were a variety of defined roles, then an ad-hoc high policy level could be generated with a quasi-SQL (Structured Query Language) call (or X-PATH). The returned record is the ad hoc high level policy that is translated by a rules engine to a low level policy. This would give an adaptability, immediacy and a universality that no other system could provide, and it wouldn't require a Subject-Matter-Expert level of knowledge. It would be embedded in the rules.

7. A multi-threaded application.

8. A JCOPS implementation and a Java SIP implementation.

9. A graphical GUI based on XML-VML (Vector Markup Language) rather than stored graphics.

10. Java RMI versus Corba. The footprint of a Java RMI is orders of magnitude smaller than a CORBA, openOrb implementation.

11. Automated policy brokerage.

12. Network topology self-discovery.

13. An intrinsic discrete set of embedded interface classes that can be quickly changed to avoid deprecated classes in future enhancements.

14. Web centric elements written in XML, eXtensible Stylesheet Language Transformations (XSLT), Portlets, Translets and various Java entities, completely eliminating HTML.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) NRNS Incorporated 4043 Carling Avenue Ottawa K2K 2A3	2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable) UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

Evaluation of Two Experimental Policy Based Network Management Systems (U)

4. AUTHORS (Last name, first name, middle initial)

Bodnar, K.

5. DATE OF PUBLICATION (month and year of publication of document) July 2005	6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.) 21	6b. NO. OF REFS (total cited in document) 2

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DRDC Ottawa/NIO Section
3701 Carling Avenue
Ottawa K1A 0Z4

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant) 15BF27	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written) W7714-3-800/001/SV
10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor) DRDC Ottawa CR 2005-108

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
(   ) Distribution limited to defence departments and defence contractors; further distribution only as approved
(   ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
(   ) Distribution limited to government departments and agencies; further distribution only as approved
(   ) Distribution limited to defence departments; further distribution only as approved
(   ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Full Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) This report compares and evaluates two experimental policy-based network management (PBNM) systems to determine the extensibility of each system with respect to the creation of a full-feature generic Policy-based Network Management System. One system is the Coalition Network Management System developed by Australia. The other is a Policy-based Network Management System developed by the University of Murcia (UMU) in Spain. The report finds that each system would require extensive development, or re-development to be properly deployed in a tactical environment. However the extensibility of the UMU-PBMNS is found to be more advantageous, easier to deploy and maintain, and to remain easily upgradeable as the scope of requirements changes.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Coalition Network Management
Network Managament
Policy
Policy-based Network Management
Policy Decision Point (PDP)
Policy Enforcement Point (PEP)
XML Policy