2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises

FCAPS in the Business Services Fabric Model

Pankaj Goyal, *Senior Member, IEEE*, Rao Mikkilineni, Murthy Ganti

Abstract—**The emergence of the next generation of computational environment, consisting of interoperable public clouds, private clouds and private infrastructures, will provide on-demand ubiquitous virtual resources. The delivery of business services over these ubiquitous virtual resources requires a reexamination of how these services are managed and delivered to assure appropriate levels of service availability, performance, security and cost. In addition, these business services maybe realized using public and private services that are able to utilize all or restricted types of virtual resources.**

The focus of this paper is the application to services of the well understood concepts of Fault, Configuration, Accounting, Performance and Security (FCAPS) management introduced in the Telecommunications Management Network (TMN). The paper proposes the incorporation of FCAPS management capabilities at a minimum into all services at a component level to enable end-to-end service management.

The FCAPS capabilities are realized through management policy, and managed through collaboration between agents implementing the FCAPS capability; a commonly used subset list of FCAPS capabilities – these can be individually or as a sub-group realized by service agents for incorporation in both managed and management service agents. The model does not mandate a centralized or distributed FCAPS management capability. The model does indicate that the management capabilities are required at every layer of the services environment.

Index Terms— **FCAPS, manageability, operability, clouds, services**

I. Introduction

The emergence of the next generation of computational environment, consisting of interoperable public clouds, private clouds and private infrastructures, will provide on-demand ubiquitous virtual resources. The delivery of business services over these ubiquitous virtual resources requires a reexamination of how these services are managed and delivered to assure appropriate levels of service availability, performance, security and cost. In addition, these business services maybe realized using public and private services that are able to utilize all or restricted types of virtual resources.

Goyal [1] presents the concept of a business service fabric (BSF) – a web of inter-connected and inter-related services, from business-oriented services to more implementation IT-oriented services and resources. The paper also presented the concept of logical and virtual partitioning of the business service fabric into sub-fabrics – islands of services. These service islands, or virtual business service fabrics (VBSF), may span company, geographical, and technological boundaries, public and private clouds, and corporate data centers – "virtual integration and partition" of the resources. Distribution of services within a BSF amongst one or more sub-fabrics (or VBSF) facilitates manageability, including security. Sub-fabric mediator services provide bridges between VBSFs, of the same BSF, manage and control inter sub-fabric interaction, manage protocols, including protocol conversions, and monitor and manage the underlying sub-fabrics [1]. In a business sense, the sub-fabric mediator services manage the interaction between, say, partners.

Current Cloud environments do not have great manageability and operability capabilities. The situation becomes more complicated in a fabric that integrates public and private clouds and private infrastructures. Business services oriented manageability and operability, including Reliability, Availability, Usability, Scalability, Performance, Configuration and Security, introduces its own challenges

Goyal *et al*. [2] show how BSF and VBSF provide a number of benefits for services-driven manageability and operability, including localization of services within sub-fabrics and isolation from services in other sub-fabrics; this makes services, including messages and resources, more manageable. The paper also introduced how the manageability and operability capabilities, including the necessary monitoring and measurement systems, may be provided by components of a service, external managers, and also the sub-fabric mediators and sub-fabric controllers that mediate inter- and intra-sub-fabric communication. It introduced the concept of business-services focused Business Service Management (BSM) that links business services to their underlying realizations.

One of the first formal service management models was developed by the International Telecommunications Union (ITU-T) in relation to its Telecommunications Management Network (TMN) that defined the general management functionality into five key areas – Fault, Configuration, Accounting, Performance and Security or just FCAPS [3]; the model was later adapted for data by the International Standards Organization [4]. Some subset of the FCAPS functionality would be performed at different layers of the TMN layers – Business Management (BML), Service Management (SML), Network Management (NML) and Element Management (EML). In the network world, FCAPS is a necessary capability provided by both hardware and software providers. Nothing inherent in the FCAPS model restricts its use only in a network environment; any network specific capabilities are easily abstracted to services.

The focus of this paper is the FCAPS manageability and operability capabilities for the services in the Virtual Business Services Fabric. The adoption of these capabilities as the minimum provided by the various clouds and private infrastructures can ease end-to-end services management. Dynamic policy-driven Manageability and Operability Process Services (DMOPS) provide fully-automated closed-loop capabilities; closed-loop because the managed services also consist of manageability and operability capabilities that provide information to the DMOPS. DMOPS are also services subject to manageability and operability. Policy-driven

P.Goyal is a principal with MicroMega, Denver, CO 80250-1496 USA (address all correspondence to pgoyal@ ieee.org).

R. Mikkilineni, is CEO of Kawa Objects, Santa Clara, CA, USA. (e-mail: rao.mikkilineni@kawaobjects.com).

M. Ganti is a principal with Rhia Systems, NJ 80309 USA. (e-mail: msganti8@gmail.com).

aspects take care of the situation where the actual environment is shared among parties with differing needs, risk profiles, etc.

First, this paper introduces the FCAPS capabilities and then later shows how these FCAPS capabilities can be implemented in a services-centric environment.

II. Basic Concepts

A. Manageability and Operability

Operability is the ability to keep a service in a functioning and operating condition – available (i.e., ready for use when needed), capable (i.e., able to perform what is needed), practicable and reliable (i.e., able to provide service continuity while in use), safe (i.e., is not dangerous to the user or the environment), and secure (i.e., preserves integrity and confidentiality). Operability also requires an inventory of the services and the ability to determine the impact of change upon the services, as well as other related and dependent services.

A service is operable if it is capable of meeting the steady-state and dynamic performance requirements in the presence of expected issues, such as non-uniform loads, excess demand, and poor performance of components or services that it depends upon. In the case of issues, the service may only be able to achieve a certain level of performance which may not be the desired performance. Operability can be restored to the desired levels by addressing the cause of the issues, for example, if the cause is excessive load by balancing load with another service instance. Operability issues maybe fixed internally by the service, externally by some other (management) service or both. In computer systems, patch management is a mechanism to address operability; the ability to install/propagate the patch expeditiously and correctly is a manageability function.

Manageability is the ability to monitor and control the service – its creation/instantiation, performance, scalability, availability, changes, risk and termination/de-instantiation. Manageability can be either purely external, or a combination of internal and external where some management capabilities are internal to the service and some are external or both; in the latter the service is said to be self-managing.

B. Service and, Agents

In the extended services model [1], processes, resources (computing, network and data), messages, people, and all other "things" are "services" (the term "business services" differentiates between the other widespread use of services). A service is composed of other services and interacts with some other set of services through messages. The implementation of a business process/application utilizes a subset of the available services subject to policy restrictions. Sub-fabrics restrict the possible interactions, the type of resources, their location, manageability and operability options. Mobility, thus, can be restricted to well-specified service implementations (agents), middleware, network segments, client devices, compute servers and data servers.

An agent realizes or implements a service; a service can have one or more realizations. A service can be realized by multiple agents where the agents have certain capabilities. For example, a Service S0 realized by agents A01 and A02, where A01 guarantees a better performance than A02. Or, A01 is an implementation that conforms to laws of country X whereas A02 conforms to laws in a set of countries. The choice of provider agent depends on requestor agent capabilities, performance, management policies, and cost considerations; policies specify required (of the requestor agent) and supported capabilities, like manageability and operability.

Policies constrain the behavior and utilization of resources, and apply to (or govern) Agents. Policies can be classified as Governance or Management policies. A governance policy imposes an obligation on the agent – a task or action that must be performed; for example, logging of all messages and usage, or the validation of the "state" at the end of a task. Capabilities and permissions are two different types of management policies. The latter permissions are about rights to perform some task, while the former is about the capability to monitor, manage faults, configuration, capacity, performance etc.

C. Service Agent Composition

Composition is a mechanism to create an object using other objects as its component. Composed objects dynamically acquire functionality at execution-time through their references to the component objects. A major advantage of this approach, over inheritance, is that the "performance" of a composed object changes as implementation of one or more of its component objects changes. Because objects are accessed only through their interfaces, one object can be replaced with another that at a minimum supports the interfaces of the replaced object.

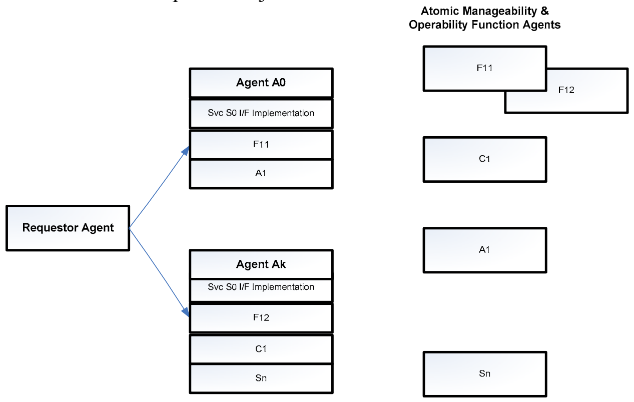
Atomic Manageability & Operability Function Agents
Requestor Agent
Agent A0
Svc S0 I/F Implementation
F11
A1
Agent Ak
Svc S0 I/F Implementation
F12
C1
Sn
F11
F12
C1
A1
Sn

Figure 1. Agents Composed with internal FCAPS Services

In this model, a service S_0 can be realized by multiple service agents, A_0 to A_k, with the implication that these service agents are interchangeable (see Figure 1). To add a particular security capability S1 to agent A_k that was initially not implemented with it, agent A_k is recomposed with S1. In Figure 1, service agent A_0 is composed of FCAPS capabilities implemented by agents F11 and A1, while service agent A_0 is composed with FCAPS implemented by agents F12, C1 and Sn; one of the FCAPS capabilities is shown to have two different realization F11 and F12 although they are the same capability. If the service agent realizing the invoking service invokes A_0 it may experience a certain level of "performance" – manageability and operability – whereas if it invokes A_k it may experience a totally different level of "performance." It should be noted that the behavior of the object with respect to the expected results for the common interface will be the same no matter which realization is invoked.

D. Service Agent Categorization

Goyal *et al.* [2] define a categorization of service agents; these characterizations are useful in managing and operating services.

Adapter agents: allow business services to use common interfaces to address other services regardless of their native interfaces. Adapter agents typically allow request for operational access to single resources (the common interfaces is to multiple resources but the access is for a resource at a time). Adapter agents do not typically support generic functions for all functions that can be performed using the native protocols; generic functions can be supported for data access.

Group agents: simplify the way a management agent accesses a group of managed agents for requests (pre-defined) sequences of operations to be performed on some or all managed agents of the group. These conceptual group agents permit generic functions to be defined that involve the same or similar operations in some or all of the agents of the group.

Orchestration agents: allow business services to interact with agents realizing elements of a service and make it possible to synchronize many different events/operations that might apply. Orchestration agents enable the performance of complex operations on a dynamic and diverse grouping of agents and control behavioral changes during operation.

E. BSF Manageability and Operability Capabilities

The BSF, VBSF and the services model provide the following:

- Management systems covering at the minimum FCAPS functionality, including self-management.
- Capability for the provisioning of services, including flexible control and management capabilities.
- Ubiquitous Virtual Services with embedded management, including self-management, capability in all services.
- Virtual resource-facing services enabling flexible usage of the physical resources.
- Virtual service clouds, “virtually integrating and partitioning” the necessary resources from public and private clouds and private infrastructures.
- Mechanisms for dynamic deployment of new management functionality through composition.
- Mechanisms for dynamic deployment of measuring and monitoring probes for service behaviors. Use of monitoring services in support of the self-management functionality.
- Mechanisms for allocation and negotiation of different resources.
- Mechanisms for self-adaptation and self-composition of resources to achieve effective, autonomic and controllable behavior.
- Increased level of service management, including discovery, configuration, deployment, utilization, control and maintenance.
- Policy driven dynamic orchestration and integration of management functions, including corrective actions.

III. FCAPS

This paper restricts itself to fault, configuration, accounting, performance and security management – in short, the FCAPS capabilities. FCAPS have been adjudged to be the basic minimum management and operability capabilities necessary to offer useable services. ITIL [5, 6] guidelines have been gaining a lot of attention and increasing use, in particular, in the areas of IT Service Management. A comparison between the ITIL IT Service Management and FCAPS would show that FCAPS has most of the critical capabilities necessary for manageability and operability; ITIL of course has lot more in the form of guidance on processes and methods.. The table below lists the various capabilities of FCAPS:

Each of these elemental capabilities may be implemented using agents that also utilize other agents such as instrumentation and monitoring agents.

By separating services into fabrics and sub-fabrics and by incorporating FCAPS capabilities within the service, a service can only affect or be affected by other services in a controlled fashion; service security enforces its own access rights and permitted actions. At each layer of our Services Fabric, there are one or more

Fault Management	**Configuration Management**
• Fault detection	• Service initialization
• Fault correction	• Service provisioning
• Fault isolation	• Auto-discovery
• Service recovery	• Backup and restore
• Alarm handling	• Resource shut down
• Alarm filtering	• Change management
• Alarm generation	• Pre-provisioning
• Fault correlation	• Inventory/asset management
• Diagnostics	• Copy configuration
• Error (fault) logging	• Remote configuration
• Error (fault) handling	• Automated software/fix distribution
• Error (fault) statistics	• Job initiation, tracking, and execution
Accounting Management	**Performance Management**
• Track service/resource use	• Utilization and error rates
• Cost for services	• Performance data collection
• Accounting limit	• Consistent performance level
• Usage quotas	• Performance data analysis
• Audits	• Problem reporting
• Fraud reporting	• Capacity planning
• Combine costs from multiple resources	• Performance report generation
• Support for different accounting mode	• Maintaining and examining historical logs
Security Management	
• Selective resource access	• Security alarm/event reporting
• Access logs	• Take care of security breaches and attempts
• Data privacy	• Security-related information distributions
• User access rights checking	
• Security audit trail log	

management agents managing the services in their domain. All communications is by messages which themselves being first-class services have self-management capabilities and are also externally managed. The sub-fabrics also reduce the amount of “management data” (for example, various performance measures) to be managed at a manageable level – the sub-fabric “partitions” the “total” raw data generated from all sub-fabrics into sub-fabric specific data only.

Next, the different FCAPS capabilities are examined in the context of the Business Services Fabric. Before visiting the various capabilities, let us look at a generic capability that is common to many of the FCAPS capabilities – events and performance instrumentation. For example, the Alarm filtering capability in Fault Management can be implemented using an Event Filtering service agent.

A. Events and Performance Instrumentation

Event correlation and event filtering: Highly scalable multiservice platforms present unique demands on existing event management systems because of the volume of traffic they process and the volume of alarms they can generate. In a virtual service environment, the events are restricted within the sub-fabric, thus, keeping the volume of events to be handled at any given location to be small and manageable. An Event Manager component within the managed service and its management service supports event correlation and filtering to reduce the potential flood of events; event filtering and correlation policies define the filtering and correlation performed. A correlation policy can be defined that links all associated events to a given root event, provided they arrive within the specified time interval. As a result, only the root event is forwarded, reducing the alarm overload on the management system.

Instrumentation: The instrumentation and management interfaces of a service are important aspects of its manageability. The service would be unmanageable if it does not have the proper instrumentation to provide information and control.

An external management system can structure and initiate a query for all instrumented measures for status or trend analysis. For high-availability, alarms in the buffer should be logged to prevent loss in case of failover or restart.

Centralized performance monitoring and trending are difficult to perform in a highly dynamic very large distributed environment. By partitioning the monitoring capability amongst the virtual fabrics, the volume of data becomes manageable; in centralized performance monitoring system the volume of data available from a large number of services is likely be too high for a performance monitoring component to collect, store, correlate, and process.

Threshold Monitoring: A Monitor component of the Event Manager monitors counters against configurable thresholds. An alert is generated whenever the threshold is crossed during some configurable monitoring collection interval; the thresholds and collection interval can be configured individually for each attribute being monitored. Different levels of alerts can be raised when an attribute has multiple thresholds; for example, a pressure monitoring system may have critically low, low, high and critically high thresholds. Rules may specify the generation of alarms after a threshold condition is met. For example, in the absence of a corrective action the attribute value may continue to be above the threshold, and the rule can either restrict further alarms being generated, or restrict their generation frequency or raise the alarm level and/or the alarm receivers.

Instrumentation is required to protect services from losses caused by security problems and to ensure instrumentation security access to that instrumentation must also be protected. New task ID-based security profiling provides more granular control of each task than typical role-based access controls; tasks themselves can be at different levels of granularity depending upon the criticality of the asset being protected. In task ID-based security, user types can be defined and then sorted into groups. Each group is associated with a particular task group – configure threshold tasks, for example – with explicit privileges (read or write). Task ID also provides flexibility in router management task authorization. To help ensure software image integrity, loadable software is digitally signed and authenticated by the installation manager during the installation process. If a package fails authentication, it is not executed.

Access to the information and control enabled by embedded instrumentation is gained through interfaces and messages.

B. Fault Management

Fault management (FM) is the collection and analysis of alarms and faults in the service. These faults can be either transient or persistent. Transient failures are not alarmed if their occurrence does not exceed a threshold; for example, sporadic message losses or delays. These events are, however, logged. Some transient problems can be automatically corrected within the service, while others may require different levels of management services to resolve. Faults can be determined from unsolicited alarm messages or by log analysis; the latter may be the only course when, say, existing services/applications do not have internal monitoring and/or alarm generation capabilities.

The FM function analyzes and filters the fault messages and coordinates the messages so that the number of actual events reflects the real conditions of the services. The root cause is reported, while suppressing other related fault messages. While all faults are logged, and an FM at some layer may have been able to resolve the fault, the resolving FM will create a trouble ticket recording the fault details and any corrective actions performed. For example, while the FM may have decided that a particular service resource R_a has failed and elected to use an alternate service resource R_b, R_a still needs to be fixed.

Firstly, some commonly used terms are defined and then the key fault management capabilities are discussed – simple capabilities such as fault logging or alarm handling aren't discussed.

1) Terms

Failure: A service S_0 is said to have a failure (not necessarily failed) if the results that S_0 delivers deviate from the expected specification for some specified period of time. The latter is important (a) so that a momentary (less than the specified duration) variances do not generate automatic correction; but also (b) so that repetitive momentary variances during some specified time window may be a predictor of forthcoming problems.

Fault/Error: is the attributed cause of the failure. A fault in the service S_0 causes failures in other services (including its components) that interact with S_0 – either a component internal to the service S_0 or an external service (the environment) that interacts with the service S_0; the fault in S_0 is a failure from the point of view of the internal component or other interacting services. A fault can lead to other faults, or to a failure, or neither. Faults have effects and the effects last for some duration of time; faults or their effects that last for a significantly long time are persistent.

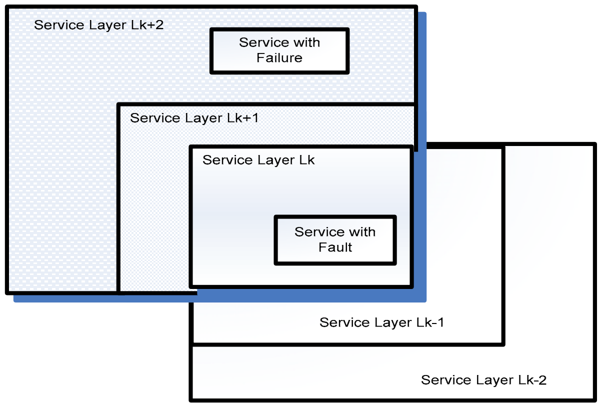

Figure 2: Fault Floor

Although all components are theoretically capable of having faults, for any service, in a particular situation, there is a level beyond which the faults are "not interesting." This level is called the fault floor. For every service, there is a mapping to the underlying services/resources and allows to trace a failure (the service where effect noted) to the underlying service/resource at fault. For example, in the case of an input message queue build-up, of an otherwise functioning service, the fault floor is the service as the fault can be corrected or mitigated by provisioning one or more service instances and distributing the load between the various copies. If, however, the output message queue is growing then the fault maybe in the recipient service(s), the messaging service, or any one of the underlying resources that implement the messaging service. The diagnostics of these various services will identify the service which needs to be addressed; so if the fault is in, say, network congestion then provisioning alternate network path would correct the problem. It should be noted that in our model, every service has manageability and operability capabilities, and as such the network fault would have been automatically addressed and would not have been the cause of the problem in our example.

2) *Service Fault Management Capabilities*

Fault Detection: The process of determining that a fault has occurred. The effects of a fault are either that the result is wrong (i.e., value fault) or that the result does not meet the performance requirements (i.e. timing fault).

Acceptance tests are the more general fault detection mechanism; comparison of the results from concurrently executing versions is an alternative to acceptance tests. In acceptance testing, the service is executed with known inputs and the actual "result" is compared against the expected "result" for the given inputs; for value faults the "result" is the outputs while for timing fault it is the performance. In an environment where multiple versions of the service are executing concurrently then the "results" from all of the versions for the same inputs are compared and the majority "result" is accepted.

Fault Isolation: The process of determining what caused the fault, or exactly which component is faulty. In the comparison method, this requires 3-or more executing version Preferable odd number of versions); majority vote.

Fault Correction: The process in which faults are removed from a service. In well designed fault tolerant services, faults are contained before they propagate to the extent that service delivery is affected. This leaves a portion of the service unusable because of residual faults. If subsequent faults occur, the service may be unable to cope because of this loss of resources, unless these resources are reclaimed through a recovery process which insures that no faults remain in service resources or in the service state.

Recovery: A number of measures can be undertaken to "recover" from faults and some of these are mentioned below.

Fault Containment: The process that prevents the propagation of faults to the extent where it can have an effect on the service to the user. It also prevents propagation where cascading faults leading to catastrophe can occur.

Fault Masking: The process of ensuring that even in the occurrence of a failure only valid results are propagated beyond services where a user may get impacted. For example, in the case of an account balance enquiry the last valid data and the date/time is presented to the user, possibly together with a message that more current data would be available at "hh:mm on mm/dd/yyyy."

Fault Compensation: If a fault occurs and is confined to a component, it may be necessary for the service to provide a response to compensate for the output of the faulty component. This is possible in certain situations, such as when reporting, say, weight where the balance component has been determined to consistently return the actual weight plus some fixed known amount.

Fault Recovery: When a service completely fails, recovery may entail restarting the service. The Configuration Manager (CM) restarts the service based on the recovery process defined for the service. The CM may provision a service recovery choreographer-service that would enforce the recovery constraints on message order, state consistency, and communicate progress to interested parties; services being stateless do not require recovery of state, however, for applications masquerading as services, state reconstitution by replay of messages form logs may be necessary.

Fault tolerance: A service built with fault tolerance capabilities will manage to keep operating, perhaps at a degraded level, in the presence of faults. For a service to be fault tolerant, it must be able to detect, diagnose, contain/confine, mask, compensate and recover from faults – that is it must have self-management capabilities.

Active Fault Prevention: When a service deviates from its normal behavior, even if the behavior continues to meet system specifications, it may be appropriate to reconfigure and/or re-provision the service to reduce risk of potential failure. For example, repetitive momentary variances in performance, while still meeting specifications, may point to the need to take fault evasion measures. Not to many years ago it was common practice to periodically (daily, weekly) reboot servers; data suggested that this significantly reduced system crashes or other server performance problems.

C. *Configuration Management*

Configuration management (CM) capabilities are responsible for the life-cycle of a service agent from its inception to final shut down; CM standardizes the activation and deactivation of services in a regulated and controlled manner. CM also includes change management to keep track of modifications to the services. Configuration information is the make-up of the service and agents that realize it. Configuration management provides the location, setup, inventory and maintenance of service agents, their components and their realization configurations. Information on the service agents is collected regularly, tracking the types of resources and their details. When changes in a service agent configuration occur, the CM can collect and analyze the changes, and ensure that these were authorized and are acceptable; unauthorized changes are reversed and also alarmed as they may be a pointer to, say, a security breach.

The configuration of multi services is complex and failure or delay can have a detrimental impact on many customer services. Configuration Manager needs to support transaction-like configuration operations where multiple features in one or more service agents can be configured at the same time in a single action; this can be achieved using Group and Orchestration agent organizations.

The CM is responsible for configuration change management. The new proposed configuration version is compared with the current/last working version and all changes are identified and notified to responsible parties. The CM can also verify if the proposed version has been used in the past and any problems that it caused; previous version that have caused problems may have been marked as unusable in which case if there is a match the new version would not be allowed. The CM maintains all past configuration versions allowing quick restoration to a previously working configuration. All changes have to be authorized and change notices sent to multiple users and service agents.

The auto service discovery tool and methods component of a CM provides continuous discovery and mapping of services, their dependencies, components and configurations with respect to their underlying realizations. The tools provide accurate, real-time visibility into the service configurations.

D. *Accounting Management (AM)*

AM capabilities deal with specifying the parameters to be monitored that define usage, setting usage limits (this is done through a configuration manager), monitoring and costing usage, ensuring that usage quotas are not exceeded, detect and report fraud (and attempts to defraud), and support accounting audits (logging). The AM also collects the accounting and usage information, analyzes the information (for example, to detect fraud, resource utilization) and sends reports to other services (for example, other AMs). The AM supports audits and fraud reporting by analyzing suspect and/or unusual behaviors. Precise accounting across all layers of the environment is required.

E. Performance Management (PM)

PM measures and analyzes service performance. PM collects resource performance data, evaluates the data and raises alerts when the actual performance is in variance from the performance threshold limits. The PM may be able to take corrective action if the scope of the possible correction is within its scope; the scope of PM of a service agent is the service agent. PM maintains logs and performs trend analysis to predict and anticipate performance problems.

The PM consists of performance policy definitions, measurement and analysis systems. Services may temporarily run short of capacity, for example, in the face of high demand. To improve the performance of a service, including capacity, the PM may cooperate with Configuration Managers (CM) to improve performance of the underlying services.

F. Security Management (SM)

SM provides multi-services, defense in-depth level security to control access and utilization of the services, maintain privacy, confidentiality and information integrity. SMs are designed to protect the services and prevent malicious, negligent and abusive behavior by authorized and non-authorized users alike. In our model, SM capabilities are distributed at all levels and the service realizations may also implement or incorporate the needed security for the protection of the service. In this defense in-depth approach inherent in our model, selective services may require multiple levels of screening of access and message flows; for example, at every mediator, collector and service itself.

An SM maintains access rights (task based), access logs, audit trails, management and governance policy enforcement, raise security alarms and distribute necessary security related information. In our model, some of the service interfaces may be secure and, thus, be under access control. The security capabilities may be configured or changed only by authorized personnel; all changes would be distributed.

Service realizations may require point-to-point and/or end-to-end security mechanisms, depending upon the degree of threat or risk. Traditional, connection-oriented, point-to-point security mechanisms may not meet the end-to-end (e2e) security requirements of services. For example, traditional network level security mechanisms, such as Virtual Private Networks (VPNs) and Secure Multipurpose Internet Mail Exchange (S/MIME), are point-to-point technologies and are not sufficient for providing e2e security for a services environment that uses messages for complex interactions and where messages flow between and across various trust domains, facilitated by intermediaries.

Therefore, e2e message-level security is important, as the intermediary may need access to some but not all of the information in the message. In this e2e security encompasses the security of messages between the intermediary and the original requester agent, and the intermediary and the ultimate receiver agent; when more than one intermediary is involved it also covers the security between each of the adjacent intermediaries. Secure message protocols have to be constructed to secure the message payload while providing access to, say, performance level information; for example, the message may specify non-logging at intermediaries or specify end of message life.

Messages to the mediators, controllers, manager components of services and management services are screened to ensure security. Virtualization provides many security benefits including:

- A well defined and multi-level multi-method securable services fabric.
- Quickly restore a malware-infected service with a clean, secure service.
- Individualized encryption mechanisms
- Isolated separate services environments.

While virtualization can have many worthwhile security benefits, security also becomes more of a management issue. In a services environment, there are innumerably more services to secure and more interconnection points. The security manageability problem is addressed by distributing security management to all levels and all service realizations, where each realization may implement a particular security capability differently; this significantly complicates the task of anyone with the intent to cause harm.

All services in the implementation of a process must meet or exceed the security requirements for the process. It should be noted that sub-processes may require a higher level of security control; the differences in the security levels require a controller or mediator service at the interaction point.

IV. Conclusion

In practice, some elements of the FCAPS model have been implemented more widely than others in coordinated management schemes. Generally, enterprises apply FCAPS to fault and configuration issues. The security function has depended on other tools that are not integrated into the overall FCAPS model. This is unfortunate, because fault, configuration and performance problems may actually be a security problem in disguise. Likewise, unusual traffic patterns captured by accounting management may indicate a security problem. The bottom line is that, although there are five elements to FCAPS, one element can influence the success of another element.

The business value of employing FCAPSis:

- Increased productivity for the IT and telecom staffs.
- The ability to manage more remote resources with little or no on-site IT staff.
- Greater user satisfaction with voice services.
- The ability to predict, and therefore prevent, performance problems.
- Potential reduction in trouble tickets returned because of faulty referral to the incorrect problem solver.
- Improved visibility into all the network resources.
- Discovery of underutilized resources that can be reduced or eliminated, resulting in cost reduction.
- Determining what resources are over-utilized and modifying their operation before users complain about the resource performance.

References

[1] P. Goyal, “The Virtual Business Services Fabric: an integrated abstraction of Services and Computing Infrastructure,” in Proceedings of WETICE 2009: 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (accepted; to be published).

[2] P. Goyal, R. Mikkilineni, M. Ganti, “Manageability and Operability in the Business Services Fabric,” in Proceedings of WETICE 2009: 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (accepted; to be published).

[3] "M.3400 TMN management functions", International Telecommunications Union, 1997

[4] "Information technology - Open Systems Interconnection - Systems management overview", ISO/IEC 10040, 1998.

[5] Office of Government Commerce (2000). *Service Support*. IT Infrastructure Library. The Stationery Office. ISBN 0-11-330015-8.

[6] Office of Government Commerce (2001). *Service Delivery*. IT Infrastructure Library. The Stationery Office. ISBN 0-11-330017-4.