Chapter 5: Experiment 1 – Management Workbench

This chapter introduces and elaborates the architectural abstractions that were utilized in the experimental system titled Management Workbench [94-95]. While there exist a significant number of architectural elements and design patterns in this system’s architecture, the focus of this discussion is on those that are potentially useful as Plug and Play constructs. The purpose of the workbench is to support a wide variety of management techniques that exist now and will evolve in the future. Accordingly, the workbench adopts a modular architecture so that the individual components can be developed independently and then composed together in order to function as a whole. This architecture enables new components to be developed independently and composed into the workbench at a later point in time. The reasons for adopting a modular architecture include catering to the complexity of the domain and addressing the uncertainty of possible future extensions.

In the context of Plug and Play, the objective of this experimental system is to “transition from a modular architecture into an extensible system architecture”, wherein a set of functionalities are encapsulated in a set of pluggable components that are attached and accessible to the system during run-time. This is achieved by moving from a tightly coupled components based architecture to a loosely coupled components based architecture. Accordingly, the hypothesis for this experiment is: “If components are dynamically attached and detached at run-time, there is no discrepancy in system performance”. The core functionality of the management workbench is the ability to support a multitude of techniques which are implemented as corresponding management tools. The methodology adopted to validate the hypothesis is to treat each of these management tools as a pluggable component. Accordingly, the goal of this experimental system is to “enable Plug and Play of management tools”.

This chapter is organized as follows: a) The section on the problem space provides an overview of the management workbench problem, b) The section on the solution space provides a possible solution that is Plug and Play compliant, c) The section on the requirements lists down the stakeholder concerns, d) The section on the quality characteristics lists down the functional and non-functional characteristics of the system, e) The section on decomposition brings forth the underlying component framework and supporting run-time, f) The next 4 sections provide an overview of the various components and their composition, and g) The last section discusses about the Plug and Play architectural abstractions of this experiment. It uses the architecture description template presented in Annex A to express the architecture description of the management workbench.
5.1 The Problem Space

As part of their work, managers need to cope with the life cycle process [96-99] of their management concerns: discover their management problems, experiment with management solutions, model them, bring them under control and delegate them to the next level of management. Traditionally, various philosophies like management by exception, management by objectives, management by strategies and management by walking around or a combination of any of these are adopted by managers to solve their management problems. There exists various management techniques espoused by leaders in the field that support these philosophies. These techniques are usually encoded in management tools in order to support managers. These tools enforce rigorous process definition involving checks and balances and serve as data keepers for management processes. They employ capabilities of software as a means to support the functionality needed by managers so that they can meet their management responsibilities. Managers use these techniques and their incarnation as management tools to arrive at solutions for their problems.

The existing management tools are designed and developed by multiple vendors, who have licensed the corresponding management philosophy, and adopt different realization approaches and best practices that are prevalent at the time of implementation of the management tool. There is no common agreement or standard between these vendors for interoperating data between the various management tools. As a result, managers are forced to operate in silos whenever they utilize management tools. Additionally, there is no consistency in the way management problem is defined or analysed or solved in the different management tools and these Systems cannot be extended any further as they are bound by the philosophies that they support. The underlying management processes vary with each implementation as well.

5.2 The Solution Space

One goal is to create a management workbench [94-95] in which multiple management tools can be plugged in and can work together seamlessly in order to address the manager’s management problems. Another goal is to support managers in formulating and resolving their management problem towards realizing their responsibilities wherein: defining the management problem involves identifying the purpose, stating the objectives, defining targets to be attained, planning and preparing the strategy to be adopted, identifying activities that need to be performed, identifying operational metrics, stating constraints and finally stating the overall effects (outcome); resolving the management problem involves
modelling responsibility devolution through the management hierarchy and responsibility enactment so that every manager in the management hierarchy has responsibilities towards the realization of which they perform tasks within the defined constraints.

An objective of this design is to bring in easy-to-use, zero-configuration, automatic discovery and release, flexible, standards-based connectivity to ad-hoc systems across a breadth of categories for a wide range of Management tools [94-95]. Ideally, in an organization, these tools form the backbone of information flow by providing platforms to integrate and coordinate management processes and sharing information across all functional levels and management hierarchies. Another objective is to support a comprehensive collaborative environment for management problem formulation and resolution, using many management aids/tools/techniques in a coordinated manner. The different toolsets are grouped according to their support in the realization of responsibilities and/or based on approaches to management practices espoused by leaders in the field [94-95]. These tools can interoperate if they adopt common ideas, structures, tags so that tasks can communicate their efforts towards responsibility realization.

5.3 Requirements for the Management Workbench

i) Managers should be able to select a specific management tool from a collection of management tools and work out solutions to their management problems in this tool

ii) Managers should be able to take information from one management tool and use it in another

iii) Managers should be able to use a collection of management tools to solve their management problem

iv) Managers should be able to work on multiple management problems at the same time

v) Managers should be able to work on multiple management tools at the same time

vi) Managers should be able to express their management problems in the various management tools. They should be able to use the management tool to arrive at an appropriate management solution

vii) Managers should be able to cut their management problems into smaller problems which can either be delegated or solved separately by the manager

viii) Various management techniques and philosophies like Management by Exceptions, Management by Walking around, Management by Objectives, Management by Metrics should be supported as part of the Management tools

ix) Managers should be able to expand the collection of management tools available at their disposal in the management workbench
x) Managers should be able to view their management solution in dashboards provided by multiple management tools

xi) Managers should be able to save the state of their management solution and resume solving it at a later point in time

xii) Managers should be able to rework their management solution from any point of their solution formulation

xiii) Managers should be able to reuse their solution approach across different management problem solving situations

xiv) Managers should be able to redefine their management problem in the management tool. They should be able to work out a management solution for this problem by either modifying the existing solution or by creating new solution

 xv) There should be a common elemental data structure/schema for the management data so that interoperability amongst the management tools can be supported easily. This data schema should be extensible so that new management tools can also be encoded

xvi) There should be a common collection of primitive graphical user interface (GUI) elements using which all the user interactions are expressed. This collection of GUI elements should be extensible so that new management tools can be supported.

xvii) There should be mechanisms to specify functionality that is desired by a management tool as part of its management process and this functionality should be extensible.

xviii) There should be a process model and an underlying notation to express all management processes. There should be a schema to express management processes. There should be mechanisms to support definition, analysis, manipulation and execution of processes.

xix) Information pertaining to a management tool can be collected in any format. This information should be converted and expressed in a unified format which is common across all management tools. It should be possible to combine multiple management tools together

xx) It should be possible to store the progress of a management process and the corresponding data collected by the management tool during the execution of the management process

### 5.4 Quality Characteristics of Management Workbench

Quality characteristics of a system are a set of essential and distinguishing attributes that have a pragmatic interpretation of the system’s inferiority or superiority. By controlling the quality characteristics, it is possible to ensure that the system delivers desired value to its stakeholders [100-
For this discussion purposes, the author utilized ISO 25010 [103] as the reference quality model. The subsequent sections list down the quality characteristics of the Management workbench.

1. Functionality:
   a. Suitability: Aids Managers in performing atleast 80% of their Management tasks
   b. Suitability: Atleast one management tool available in the workbench can be used by the Manager to solve their management problem
   c. Accurateness: Managers should be able to always performs (100%) the right task corresponding to the right technique at the right time in order to solve their management problem
   d. Accurateness: A wizard guides the Manager through the problem solving process
   e. Interoperability: All the Management tools are interoperable. Managers can transfer data from one tool to another.
   f. Interoperability: Information (regarding management problem and its solution) can be exchanged between multiple management workspaces
   g. Compliance: Complies with all the Management policies advocated by the Management Technique supported by the management tool
   h. Security: Manager can be assured that atleast 80% of their management problem can be solved by adopting the management technique encoded in the management tool
   i. Security: Uses open format for storing information in storage media and while exchanging information between multiple tools.

2. Reliability:
   a. Maturity: Management tools should be mature enough to help address 80% of the management problems faced by Managers
   b. Fault Tolerance: Any issues that arise due to interpretation of a management technique should be handled gracefully.
   c. Recoverability: On exceptions, uses rules and conditions to rollback or abort tasks (50%)
   d. Recoverability: Store and Retrieve management problems and their solutions from a storage media

3. Usability:
   a. Understand ability: Managers can comprehend and use entire system (100%)
   b. Learnability: Guides through the steps of using the Management Tool (50% guidance)
   c. Operability: Uses familiar interfaces for interacting with the User (80% compatibility)
d. Operability: Undo/Redo last few actions performed by the user on the Workbench

4. Efficiency:
   a. Time Behaviour: Manager should be able to solve their management problem within a predefined time frame. The solution should not be processing forever.
   b. Resource Behaviour: Workbench should use minimal resources to satisfy the problem solving needs of the Manager.

5. Maintainability:
   a. Analysability: Managers can keep track of progress always (100%)
   b. Changeability: Managers can change their problem definition and solution formulation on demand (100%)
   c. Changeability: Managers can solve the same management problem in multiple management workspaces
   d. Stability: Absorbs 80% of perturbations introduced by Manager and their environments
   e. Testability: System performance and its use to solve a management problem is reproducible (100%)

6. Portability:
   a. Adaptability: Adapts to change in the working environment (100%)
   b. Installability: It should be easy to expand the collection of Management tools that are available in the Workbench (90%)
   c. Conformance: Conforms to Management policies (100% compliance)
   d. Replace ability: Managers can replace any of their Problem definitions or Solution formulations (100%)

5.5 Decomposition of the Management Workbench

Management is the process of dealing with or controlling things or people and utilizing available resources in order to ensure that the organization’s commitments are met [96-99]. While the functions of a manager are predominantly planning, organizing, influencing and controlling; a manager’s main responsibility is to reduce/eradicate risks. It is therefore necessary for the manager to understand the different issues, in achieving the delivery schedule and desired outcome quality, which are potential risks and introduce appropriate interventions to mitigate these risks. In other words, it is the responsibility of the manager to identify the right problems that prevent prior commitments from being honoured and solve these problems using minimal organization resources and minimal interventions.
Since the objective of the management workbench is to support the manager in performing their
tasks, it is necessary for the workbench to support problem identification, problem formulation and
problem resolution (solutioning). The decision as to when and how to intervene is based on facts/data
that are available in the situation, hence it is necessary for the workbench to support collection, analysis
and management of management data/facts. There are many best practices and philosophies that
provide a systematic approach for problem identification and resolution. In most cases, based on the
situation to be tackled, the time and resources that are available at the manager’s disposal and the
manager’s prerogative, a particular technique/approach is utilized. Therefore, the management
workbench should be able to support multiple management techniques with the provision to allow
managers to select the appropriate technique as per their needs.

Management Workbench supports managers in formulating and resolving their management
problem towards realizing their responsibilities. Defining the management problem involves identifying
the purpose, stating the objectives, defining targets to be attained, planning and preparing the strategy
to be adopted, identifying activities that need to be performed, identifying operational metrics, stating
constraints and finally stating the overall effects (outcome). Resolving the management problem involves
modelling responsibility devolution through the management hierarchy and responsibility enactment so
that every manager in the management hierarchy has responsibilities towards the realization of which
they perform tasks within the defined constraints. It supports a comprehensive collaborative
environment for management problem formulation and resolution, using many management
aids/tools/techniques advocated by leading practitioners in the field in a coordinated manner so that
management of work can be supported.

Accordingly, the Management Workbench can be decomposed into the top level components: a)
Management component framework that caters to multiple management practices and processes, b)
Management component framework run-time that supports the Management Component framework
and also enables manipulation of the different management processes and their associated services, c)
Management Tool container which serves as a single interface to the Management tools that supports
specific management techniques and processes and the Management tool run-time that bridges
Management component framework run-time with the various Management tools and provides the
necessary Plug and Play capabilities, d) Management Cubicle is a placeholder for facts, problem situation,
managerial solution, data and causes relevant to management problems and their corresponding
solutions. It facilitates handling of multiple management problems, dealing with multiple facts, and
arriving at multiple solutions to different management problems. This decomposition is illustrated in Figure 5.1.

**Figure 5.1: Decomposition of Management Workbench**

### 5.6 Management Component Framework

Being an effective manager requires experience in handling situations, experience in handling people and experience with different management techniques. Management techniques are methods of managing that help to develop a productive workplace. There is no single management technique that works in all situations, which is why it is important to become familiar with more than one. Typically, a management technique would comprise of: a) a set of processes that the manager can utilize to solve the
management problem, b) a set of resource requirements for solving the management problem in a particular way, c) set of information structures and information flows that connect the processes with the organizational resources, and d) management work products and work items that emanate from the processes.

![Management Component Framework Diagram](image)

**Figure 5.2: Software Decomposition of Management Component Framework**

The Management component framework [MCF] is designed to support the management techniques, management process definition, process modelling and process enactment. To support management process definition, the MCF should support resource identification, resource allocation and monitoring and resource management. Management cannot happen without information. Availability of information enables manager to make informed decisions hence, it is necessary for the MCF to support information structures, flows, gathering, sensing and analysis which collectively provide the ability to make decisions. Finally, the outcome of a process is a work product. A process operates on its work-items to create these work-products. Therefore, the MCF should support work products creation and management which in this case would be management components creation and management.

Accordingly, the MCF, as illustrated in figure 5.2, provides the underlying structure in terms of interfaces, resources and protocols for the composition of components playing specific roles in the Management workbench. It is a composition of: a) Resource server that provides the necessary resources for the management processes to be executed, b) Process server that enables multiple process models to be manipulated and executed, c) Information Server that enables interoperation and integration of data and processes between components, and d) Components server that stores the various management components in an organized way and facilitates tracking of these components through its life-cycle,
manages interaction between components and external environment, specifies the way these components can be composed together and provides a means of common disposal of components that are no longer needed.

5.7 Management Component Framework Run-Time

There are many management techniques and tools that the manager utilizes to understand and address their management problems. All these techniques are nearly independent of each other and continuously evolve. A management workbench that supports all these techniques becomes too complex to design as a single monolithic system. Modularity and extensibility are design principles that facilitate extension of the workbench so that new tools and techniques that are demanded by the managers can be added to it as and when demanded. However, the class of extensions that can be supported at runtime are anticipated extensions which are predominantly variability of the parts along certain dimensions. Hence, it is necessary to identify those parts of the system that can be independently designed, segregate those parts, design them separately and then compose them into the workbench during run-time. In the case of the management workbench, the management techniques and their incarnations as management tools are candidate parts. Since the techniques that are encoded into these parts are essential capabilities for managers, a good design practice to support such modularity is to embed the necessary extensibility services in the run-time system. The objective of the run-time system is to provide necessary interfaces that support interaction with the parts, sharing of information with the parts, integration of the parts with the management workbench and management of the parts life-cycle.

A design pattern used in embedded systems to support sharing of information is the bus. The bus connects the major components of the embedded systems and allows communication of information between the connected systems using address resolution. This design pattern is a candidate pattern in the Management component framework run-time [MCFR] for supporting information sharing between the parts. Another design pattern that is used in USB devices for integrating parts is the host controller. The host controller facilitates interaction between the USB based devices and the host system. The host controller is a candidate pattern in the MCFR for supporting interaction between the parts. Another design pattern used in system integration to support information exchange is the interoperability bus. This bus facilitates conversion of information in one form into another. The bus is a candidate pattern in the MCFR for supporting information exchange between the parts. In the electrical industry, Plugs and sockets are design patterns that allow electrically operated devices to be connected to the power supply.
While the plug is a movable connector attached to the device, the socket is a fixed connector that is attached to the power supply.

The plug and sockets are candidate design patterns that can be used in the MCFR to attach parts to the management workbench. In networking, network hubs are design patterns that are used to connect multiple devices and make them act as a single network segment. It facilitates management of the device connections to the network segment. The network hub is a candidate design pattern that can be used in the MCFR to manage the connectivity of the different parts. In the Windows operating system, applications are event-driven. The various applications do not call a function to obtain the events (keyboard, mouse, display etc), but rather wait for events to be posted onto their respective message queues. This eventing capability is a candidate design pattern for the MCFR in order to handle the various events that occur in the parts that are plugged into the management workbench.

Accordingly, the MCFR provides the underlying structure to plug-in parts using the sockets, exchange information using the bus, manage the parts using network hub, handle events using the appropriate eventing system, and share information using information bus. As shown in figure 5.3, it is a composition of: a) Plugin Socket which serves as the connection point for a management tool facilitated through its run-time into the management workbench, b) Host Controller which serves as the node that...
provides resources, capabilities and services to the management tools so that they can work together as part of a network, c) Network Hub which provides resources, capabilities and services for multiple management tools to be connected and networked, d) Information Bus which provides resources, capabilities and services for manipulating and transforming the data, e) Event Manager that serves as the placeholder for receiving and processing notifications from the different management tools, and f) Event Handler which serves as the placeholder for invoking services/functionality from various management tools and processing their results in response to the various events triggered by the management tools.

In the management workbench, the MCFR provides the necessary functionality for supporting composition of components, communication between components, manipulation of components, management of components performance, invocation of functions and services supported by the components and transformation of data from one form to another. While the Management component framework provides the logical structure for facilitating components composition, the associated run-time enables extending the container and managing the appropriate resources necessary for this extension. It provides the capability of a socket that can serve as a host in a plug and play setup. Specifically, the run-time helps managers dynamically discover and attach different management tools to the workbench. It aids in identifying the underlying management processes, supported capabilities and manipulation of the corresponding data structures. The plugged in management tools can exchange data with each other using the run-time as the integration bus. The run-time utilizes standard protocols and object calling conventions to enable this seamless integration.

### 5.8 Management Tools

A management tool is an information system that is organized for collection, organization, storage and communication of information pertaining to management. While there are different areas of concerns and responsibilities for managers in their organizations, the specific focus of management tools is to provide efficiency and effectiveness of management decision making. These tools aid managers by compiling information from a wide range of sources and use this information to make informed decisions about a particular management problem that they face. Using these management tools, managers convert an ill-structured management problem into a structured management problem so that they can have clear goals, solution paths and plans to meet these goals.

Management tools are enablers of decision and action with respect to addressing a managerial problem and provide the functionality necessary for solving management problems. Every management
tool comprises of a specific management process that is adopted by the manager to arrive at solutions to their management problems. Each tool represents the management information in a specific way. The way information is gathered and the different GUI elements are used to capture this information by each of the management tools is also different. The computations that are performed on the management information depend on the technique adopted by the tool.

A design pattern that is used in OSI model of computer networking [104] is the presentation layer. The presentation layer prepares the information to be appropriately displayed in the application that the user interacts with. The presentation layer is a candidate design pattern for the management tool for presenting information. A design pattern that is used in statistics is the representation layer. The representation layer facilitates graphical representation of data and their relationships so as to draw meaning from the data. Multiple representations of the same data are often used to obtain a different perspective of the situation from which the data is collected. Representation layer is a candidate design pattern for the management tool to represent management information.

In software development environments, a workspace is a placeholder for a collection of files and resources of a project that aids the developer to work with these files and resources as a cohesive unit. A workspace typically provides a working environment to a user and facilitates their working style in order to provide a productive environment for their role based tasks. The workspace is a candidate design pattern for the management tool to facilitate the manager’s working style. In organizations, workflows are repeatable patterns of activities that facilitate business operations. They are the building blocks that
utilize the organization’s infrastructure and resources to perform and monitor the sequence of business tasks. Hence, the workflow is a candidate building block for the management tool in aiding the managers to perform their management functions.

The management tools, as shown in Figure 5.4, is a decomposition of: a) Management workspace that serves as the placeholder for performing management tasks related to multiple management problems, b) Representation workspace that serves as the collector and the placeholder for facts, data and other information related to the specific management problem, c) Management workflow that serves as the placeholder for monitoring management problem solving process progress, and d) Presentation Layer that provides the necessary capabilities and services to the management tool in order to gather information from managers.

5.9 **Management Tool Run-Time**

The Run-time, as shown in figure 5.5, provides functionality and resources needed by the management tool. It provides the necessary infrastructure for realization of the specific management technique; basic services (Garbage collection, Process management, Input/Output, Exception handling etc.) necessary for the various components to work together; supports extensibility of some kinds of components and their associated services; and coordinates management activities corresponding to the underlying management process.

It generates managerial events, when certain process steps in the management process are executed, which are then passed onto the management component framework for further action. Such events are also generated when some pre-defined conditions are met or some managerial decisions are taken as part of the management process. It provides services, data structures and functionalities for interacting with the Management Component framework and provides the plug interface for binding the management tool to the Management component framework. Using this plug interface, it becomes possible to convert a management tool into an on-demand component resource that can be manipulated and dynamically attached/detached from the management workbench.
5.10 Discussions

The experimental system addresses its goal of enabling plug and play of management tools by:

a) Encapsulating every management technique as the management tool component.
b) Encoding the plug and play interface mechanism as socket and compatible plugs.
c) Encapsulating the plug capability as part of the management tool run-time component.
d) Encapsulating the socket capability as part of the component framework run-time component.
e) Dispatching the events to the appropriate handler by using the plug & socket interface.
f) Providing the desired functions to the framework by using the plug & socket interface.
g) Managing the data to be manipulated by using the exchange services of the plug & socket interface.

The three critical elements, events, functions and data that are required for proper functioning of the management workbench have been made accessible through the plug and socket interface. Hence, it can be observed that “plug and play architectures did not impact interface performance”. Accordingly, the hypothesis “If components are dynamically attached and detached at run-time, there is no discrepancy in system performance” was verified. Further, the underlying principles and constraints that are discussed in the subsequent sections enable the transition from modular architecture to extensible architecture.
5.10.1 Principle of independent partitioning and low complexity

According to Maier [51], while partitioning a system, it is necessary to choose the elements so that they are as independent as possible. In the case of the management workbench, the management tools are independent of each other and serve as the extensible component. Each of the management tools exposes interfaces necessary to express the problem and the resulting solution, while the actual technique is not exposed to the component framework thereby reducing the complexity of the interfaces. The internal complexity is due to the steps involved in identifying the appropriate solution for the problem.

5.10.2 Principle of grouping and separating

According to Maier [51], while partitioning a system, it is necessary to group those elements that are strongly related to each other and separate elements that are unrelated. In the case of the management workbench, the resource server manages the resources, the information server manages the information, the process server manages the processes that are used for management and therefore all these are grouped together as part of the management component framework. However, the management tools support individual management techniques and hence they are separated as pluggable components.

5.10.3 Principle of minimum communication

According to Maier [51], while partitioning a system, it is necessary to segregate into elements that have minimum communication amongst themselves. Accordingly, while each of the plugged management tools provide functions necessary for the management component framework, they do not provide any functions that can be consumed by a different management tool.

5.10.4 Principle of accommodating change

According to Maier [51], in large systems, evolution is a process of ingress and egress. Accordingly, evolution in the case of the workbench is made possible by the ability to plug a management tool; to plug an upgraded or new tool; and to unplug a management tool that has outlived its expectations.

5.10.5 Principle of compatible support elements

According to Maier [51], the architecture of the supporting element should fit into the system that it supports. In the case of the workbench, the plug interface of the management tool and the socket interface of the component framework are designed such that only compatible tools can utilize this interface.
5.11 The Plug and Play Architectural Abstractions

This section introduces and elaborates the plug and play architectural abstractions of the management workbench. These abstractions are considered, based on the premise that “Software Plug and Play is a collection of standards, encoded in the run-time and embedded in the component schema, in order to support functional and structural extensibility of the management workbench”. These abstractions are:

5.11.1 Plug

A plug comprises of a collection of software resources that enables the plugged component to be executed on an underlying computing system. It serves as the gateway for the events, data and services that the plugged component manipulate as part of its computation. The plug encodes the interface mechanism for connecting to a compatible socket thereby enabling the plugged component to function as an external component of a component framework.

5.11.2 Socket

A socket is a handle that allows the plugged components to utilize the data, events and services that are exposed by the component framework. It facilitates two way communication of events, data and service invocation between the plug and the component framework. As an interface, the socket performs a type checking for compatible plugs to validate the connection and disconnects those that fail.

5.11.3 Component Framework

A component framework defines specific interaction and composition standards that should be adopted by the constituent components and provides the set of elements required for supporting the execution of these components. It implements a component schema which describes the characteristics of the constituent components in terms of the services, service operations, events, and other metadata that it should support.

5.11.4 Plugged Component

It extends and adds functionality to the component framework thereby enabling the evolution of the underlying system. It utilizes the run-time services provided by the plug to attach itself to the component framework.
5.11.5 Event
An encapsulated set of software actions that are triggered during component execution, along with a set of data that capture the computational state under which the event was triggered. Such events are triggered when a set of pre-defined execution conditions are met. They can be triggered inside the component framework and dispatched to the plugged component for processing or triggered inside the plugged component and dispatched to the framework for processing.

5.11.6 The Plug and Play Component Model
The Plug and Play component model is put together by considering the plug and play architectural abstractions that were discussed earlier. Accordingly, the constituent elements of The Plug and Play component model are: a) The component framework which provides the composition semantics, the logical structure for facilitating components composition, underlying structure in terms of interfaces, resources and protocols for the composition of components playing specific roles, b) The component framework run-time which serves as a socket and provides the necessary functionality for supporting composition of components, communication between components, manipulation of components, management of components performance, invocation of functions and services supported by the components and transformation of data from one form to another, c) The repository which hosts the information that can be manipulated, processed and transformed by the plugged components, and d) The components container which is a logical structure that hosts the components that are plugged in along with their respective run-times. Figure 5.6 provides a decomposition of this model.

While the plugged component hosts the services and functionalities necessary for the composed component, its run-time provides functionality and resources needed by the component, and also the ability to serve as a plug. It provides basic services necessary for the various components to work together, generation of events which are then passed onto the component framework for further action, a connection receptacle that handles the dynamic connection between the component framework and the plugged component and so on. While the component framework hosts the functionalities and processes that are utilized by the end-users, its run-time provides functionality and resources needed by it, along with the ability to serve as a socket. It provides basic services necessary for various plugged components to work together, events orchestration and so on.
This chapter introduces and elaborates the architectural abstractions that were utilized in the experimental system titled Management Workbench [94-95]. The approach to transition from a modular architecture into an extensible system architecture was illustrated by utilizing a set of pluggable components that are attached and detached to the workbench on demand. As a result, the hypothesis “If components are dynamically attached and detached at run-time, there is no discrepancy in system performance” is verified. Understanding the key architectural principles of this experiment helped identify the underlying plug and play architectural abstractions. This was culled out and expressed in the form of an abstract Plug and Play component model.