An introduction to the IBM Views and Viewpoints Framework for IT systems

By
Denise Cook, Software Engineer
Peter Cripps, Senior IT Architect
Philippe Spaas, Executive IT Architect
IBM Corporation

December 2007

Contents

Abstract

In 1995, Philippe Kruchten of Rational Software Corporation published his now-famous paper, "The '4+1' View Model of Software Architecture" (see below in Citations in this article for references). It presented a model for describing the architecture of software-intensive systems based on the use of multiple, concurrent views that allowed the

concerns of various stakeholders of the architecture to be addressed. Since the publication of that paper, many of its ideas have been included in development processes, such as the IBM® Rational Unified Process® (RUP®), and standards, such as *IEEE 1471: Recommended Practice for Architecture Description of Software-Intensive Systems*.

Background and current context

In 2006, IBM® Rational® and IBM Global Business Services initiated a project to combine the major methods used and licensed by IBM under the umbrella title of IBM® Unified Method Framework (UMF). This framework will form the basis of IBM's current and future methods, and it is based on the Object Management Group (OMG) standard for describing methods, the *Software Process Engineering Meta-Model*, commonly referred to as SPEM (see SPEM06).

To provide consistency across UMF-integrated methods and to properly define the models to support system description, it was necessary to define a standard approach for identifying, describing, and organizing the underlying method content. This standard approach would provide a framework within which the system description could be organized and also provide a set of conventions for notation, terminology, and semantics to be used in these descriptions.

The existing *IBM System Description Standard* (see Spaas), or SDS, defines such a set of conventions. Built on the IEEE 1471 standard for architecture description, it supports the separation of stakeholder concerns and provides a mechanism of abstraction that makes clear what information is needed to address those specific concerns.

The IEEE 1471 standard is built upon Philippe Kruchten's original concept of using views to address the concerns of various stakeholders of a software architecture. IEEE 1471 standardized the definition of a *view* and introduced the concept of a *viewpoint*. Although the views described by Kruchten laid the foundation for the concepts that underpinned the IEEE 1471 standard (he was a member of the working group that produced the IEEE 1471 standard), the definition of a view in the 4+1 view model is less specific and could correspond to these IEEE definitions:

- **Viewpoint:** A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

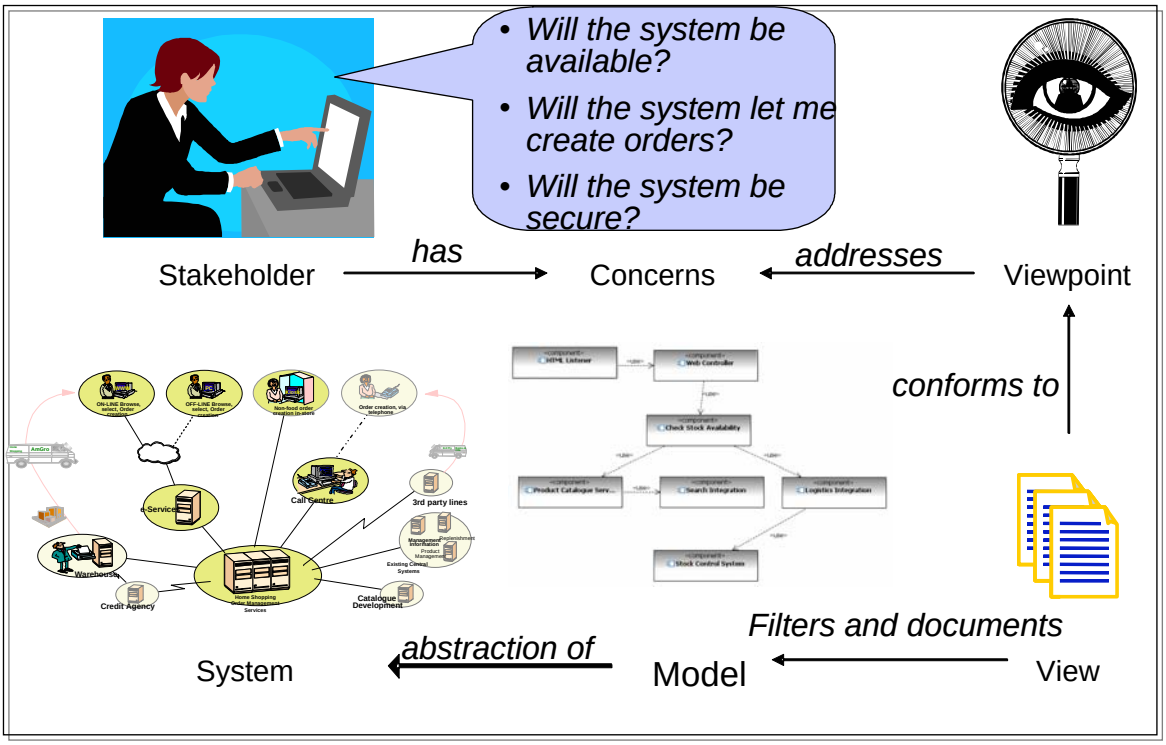
- **View:** A representation of a whole system from the perspective of a related set of concerns.

A viewpoint is basically a specification that describes all you need to do to create a particular view of your system. A viewpoint is defined with a particular stakeholder or set of stakeholders in mind. Thus, for example, a stakeholder who is interested in the

performance of the system (a Performance Architect) would have a viewpoint that would define all of the required information about performance. This viewpoint might specify how to write performance use cases and how to construct a performance model.
A view is the actual architecture description itself (in the context of a related set of concerns) that is created according to a viewpoint. Therefore, using the previous example, the performance model artifact would be considered a view of the functional and operational models, with only the performance-related elements from the system being a part of the view.

Figure 1 illustrates how viewpoints and views can help address stakeholders' concerns. It shows how we construct models, which are abstractions of the systems that we are trying to build. A model is documented through a view, which t conforms to a viewpoint that addresses the stakeholders' concerns.

Figure 1. How viewpoints and views address stakeholder concerns



Clearly, a unified method framework needs to have a consistent library of viewpoints that can be applied in a broad range of contexts. It is also necessary to create a framework, so that this library and its associated method elements can be extended and tailored as required by its users. The framework, viewpoints, and views described in this article provide a basis for identifying, defining, and organizing the system model elements and for allowing different stakeholders to focus on their own concerns.

Note:
The system model is the model of the "system in the real world," and it is the ultimate owner of all model elements and model element relationships to which all models refer. Although the principles underlying the viewpoint framework are applicable at the generalized system level, the viewpoint library described in this article is specific to the IT system level.

Viewpoint framework

It is impossible to create a single comprehensive model for a complex system that captures all of its functionality and quality properties and can also be understood by all of its stakeholders (see Rozanski, Nick and Woods, Eoin). Therefore, the viewpoint framework partitions the description of a system into a number of separate but interrelated views that, collectively, describe the whole system.

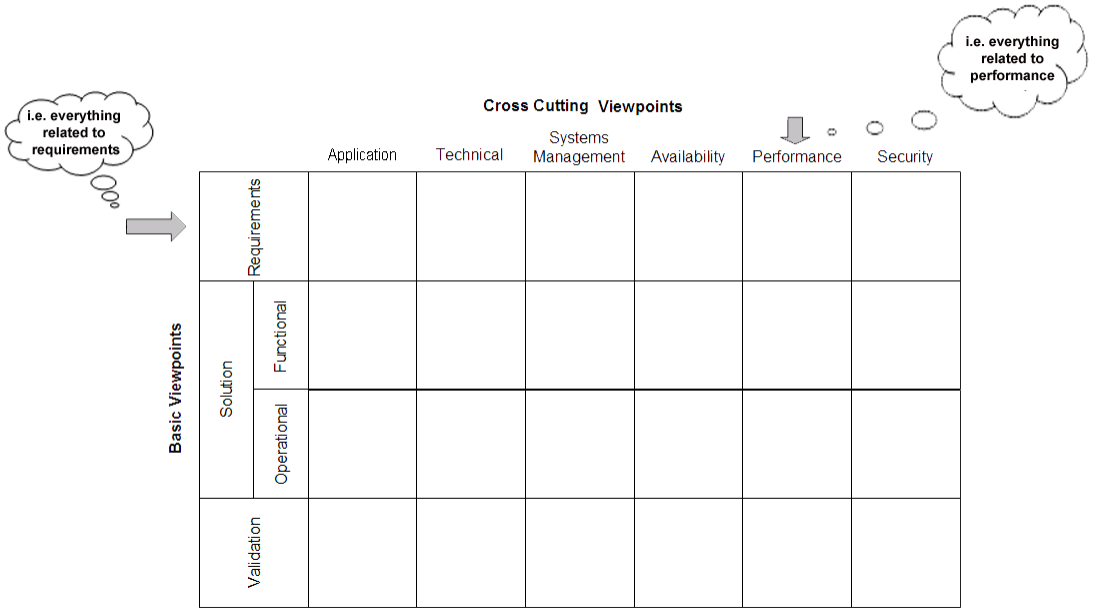
This framework consists of a library of basic viewpoints. When those associated views are combined, they form a representation of the whole system model. Basic viewpoints are focused on fundamental areas of concern that are applicable to all ways of looking at a system (for example, everything about the system related to its requirements). They allow you to reason independently about a particular, fundamental aspect of the system.

The basic viewpoints are augmented by cross-cutting viewpoints that address all possible information within that cross-cutting context. Cross-cutting viewpoints address stakeholder concerns, and, in many cases, are associated with the quality properties of a system. For example, a security cross-cutting viewpoint could filter security *requirements* from a requirements basic viewpoint but filter security *components* from a functional basic viewpoint.

Basic viewpoints focus on fundamental areas of concern about a system, while cross-cutting viewpoints define specific ways of looking at a system considering all possible information in that context.

Figure 2 provides a visual representation of basic and cross-cutting viewpoints within the framework.

Figure 2. A framework of basic and cross-cutting viewpoints



IT System Viewpoint library

Although the framework provides a way of partitioning the description of a system by using the concepts of basic and cross-cutting viewpoints, the actual viewpoints used in the framework are defined in a viewpoint library. (The terms basic viewpoint and aspect can be used interchangeably.)
The IBM IT System Viewpoint library includes the following basic viewpoints:

- **Requirements:** Model elements that capture all of the requirements placed on the system, including business, technical, functional, and nonfunctional requirements

- **Solution:** Model elements that capture the solution that satisfies these requirements and constraints, further organized into these two categories:
    - **Functional:** Focuses on the model elements that are the structural elements from which the system is built and their relationships (both static and dynamic)
    - **Operational:** Focuses on how the target system is built from the structural elements modeled in the functional view and how it integrates into its environment

- **Validation:** Model elements related to assessing whether a system will deliver its intended functionality with the expected quality of service

There are, potentially, any number of cross-cutting viewpoints. For example, the IBM IT System Viewpoint library includes these typical viewpoints.

- **Application:** Addresses the concerns that business stakeholders may have about how the system meets the needs of the business

- **Technical:** Addresses the concerns that technical specialists such as infrastructure and middleware architects may have about what hardware and software products and packages will be used to support the application-dependent components

- **Systems Management:** Addresses the concerns that systems management and operations staff may have about how to operate the system after it is deployed

- **Availability:** Addresses the concerns of architects who are interested in achieving the stated availability requirements

- **Performance:** Addresses the concerns of architects who are interested in achieving the stated performance requirements.

- **Security:** Addresses the concerns of security specialists who are interested in achieving the stated security requirements

Basic viewpoints

Basic viewpoints address fundamental areas of concern about the system. They are supported by one or more "complete" basic models (complete in that they address all possible cross-cutting concerns). For example, the Requirements basic viewpoint would specify the conventions and notation for the development of a use-case model that

refers to all of the use cases for the system, spanning business purpose, security, availability, and other cross-cutting concerns.

This section describes each of the basic viewpoints included in the IBM IT System Viewpoint library:

- Requirements viewpoint
- Functional viewpoint
- Operational viewpoint
- Validation viewpoint

Basic viewpoints are supported by complete basic models. These models are complete in the sense that they address all possible stakeholder concerns associated with the basic viewpoint. By filtering these complete basic models, you can arrive at partial basic models that address the concerns of a particular stakeholder and, hence, support cross-cutting viewpoints. Both complete and partial basic models reference model elements (and model element relationships) that are owned by the System model.

The key concepts appearing in the complete basic models associated with the basic viewpoints are identified in the following section. Notice that the concepts have been associated with the basic viewpoints that they are most closely identified with through custom and practice, and that model elements can be referenced by all basic models when they are useful in those models.

Requirements viewpoint

The Requirements viewpoint describes all of the requirements and constraints placed on the system. Requirements are either functional or nonfunctional (relevant to either the qualities that the system has or the constraints imposed on the system).

Table 1 shows the key elements of the Requirements viewpoint.

Table 1. Key elements of the Requirements viewpoint

Element	Description
Actor	Describes a role that a user type or an external system plays with respect to the target system
Use case	An identifiable and externally observable behavior of the target system or a part of the target system
User type	A person who interacts with the target system
Nonfunctional requirement	A quality requirement or constraint that some part of a system must satisfy

In principle, a complete basic model for the Requirements viewpoint (for example, a use case model) would contain all of the use cases for the system.

Functional viewpoint

The Functional viewpoint describes the structural elements from which the system is built and their relationships (both static and dynamic). Table 2 shows the key elements of the Functional viewpoint.

Table 2. Key elements of the Functional viewpoint

Element	Description
Component	A modular part of a system that encapsulates its content, with a manifestation that is a replaceable within its environment. A component defines its behavior in terms of provided and required interfaces. (UML05)
Interface	Specifies one or more operations and is both offered and used by a component.
Operation	Defines the message that can trigger a specific behavior of a component.
Data type	Provides a definition of a named set of attributes.
Interaction	Identifies the messages exchanged between one or two components in the context of a collaboration.
Collaboration	Captures the exchange of messages between components in the context of a particular scenario.
Message	Defines a single communication between the participants of an interaction.
Locality	A modular unit of functionality, which reflects that its functionality will be jointly distributed but without indicating an exact geographic location. This functionality is made available through interfaces.

In principle, a complete basic model for the Functional viewpoint (for example, a component model) would contain all of the components of the system.

Operational viewpoint

The Operational viewpoint describes how the target system is built from the structural elements modeled in the Functional view and how the target system integrates into its environment. Table 3 shows the key elements.

Table 3. Key elements of the Operational viewpoint

Element	Description
Node	A collection of components that fulfill a specific responsibility with a certain quality of service within the overall system.
Location	A geographical area or position (defines the places where users work and nodes are located)
Zone	An area for which a common set or subset of nonfunctional requirements or characteristics can be defined
Boundary	The transition between two zones associated with a change in value for a particular nonfunctional requirement or characteristic
Border	The connection between two locations
Connection	Supports the connectivity between two or more nodes
Deployment unit	An abstraction of a component or data created to simplify the placement process

In principle, a complete basic model for the Operational viewpoint (an operational model, for example) would contain all of the operational elements of the system (nodes, zones, and so on).

Validation viewpoint

The Validation viewpoint describes the concepts related to assessing and capturing the outcome of whether a system will deliver its intended functionality with the expected quality of service. Table 4 shows the key elements.

Table 4. Key elements of the Validation viewpoint

Element	Description
Risk	A risk is a potential event or future situation that may adversely affect the project.
Issue	An issue is a generic term for a matter of concern on a project.
Finding	A finding is the result of an investigation.

Cross-cutting viewpoints

Cross-cutting viewpoints address the concerns that different stakeholders have about the various qualities of the system. They Cross- define views that filter according to the information that the different stakeholders are interested in, and they are meant to answer these kinds of questions:

- Will the system meet the performance requirements of the users?
- Will the system be available at the right times of the day?

The IBM IT System Viewpoint library includes six cross-cutting viewpoints, but it can be extended as required. Each of the cross-cutting viewpoints has a different impact on the basic viewpoints. By applying a cross-cutting viewpoint to a Basic viewpoint, we hope to draw new insights into the system. Some of these insights will be captured as new artifacts that provide stakeholders with the additional information they need to address their concerns. The remainder of this section covers the impacts that the cross-cutting viewpoints have on the basic viewpoints and what new insights might be obtained by applying them.

Application cross-cutting viewpoint

The Application viewpoint is concerned with those components and their relationships which provide the application-dependent behavior of the solution (that is, behavior that the system must provide in order to meet its business requirements).
Table 5 shows the impact that this cross-cutting viewpoint has on the Basic viewpoints.

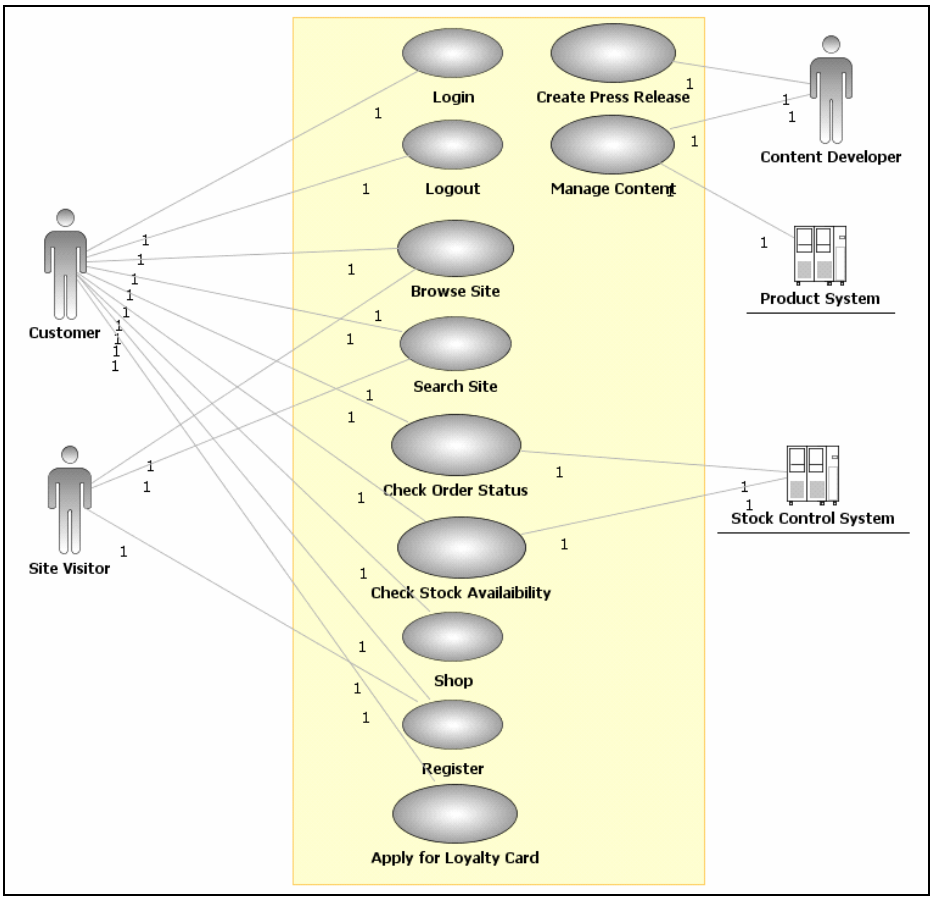
Table 5. Impact of the Application cross-cutting viewpoint on the basic viewpoints

Basic viewpoint	Impact
Requirements	This is where the functional requirements of the system are captured. One way of capturing requirements is through use cases. Here, we would capture those use cases that define the requirements of the system from a business user's perspective.
Functional	The Functional view is where the static and dynamic application-level behavior of the system is captured. It includes component and data models that address the functional requirements of the system.
Operational	The Operational view describes how the application-dependent components and data are deployed on the hardware nodes, as well as the connectivity between them.
Validation	The focus of this view is validating that the proposed solution meets the business requirements.

Examples

The use-case model shown in Figure 3 is for the Lifestyles is US (LiUS) Home Shopping System. This example will be used throughout the paper to illustrate its concepts. While this example is based on an actual customer engagement, it has been significantly reduced in scope so that important concepts can be explained without having to involve the reader in too much business detail.

Figure 3. Use-case model for LiUS

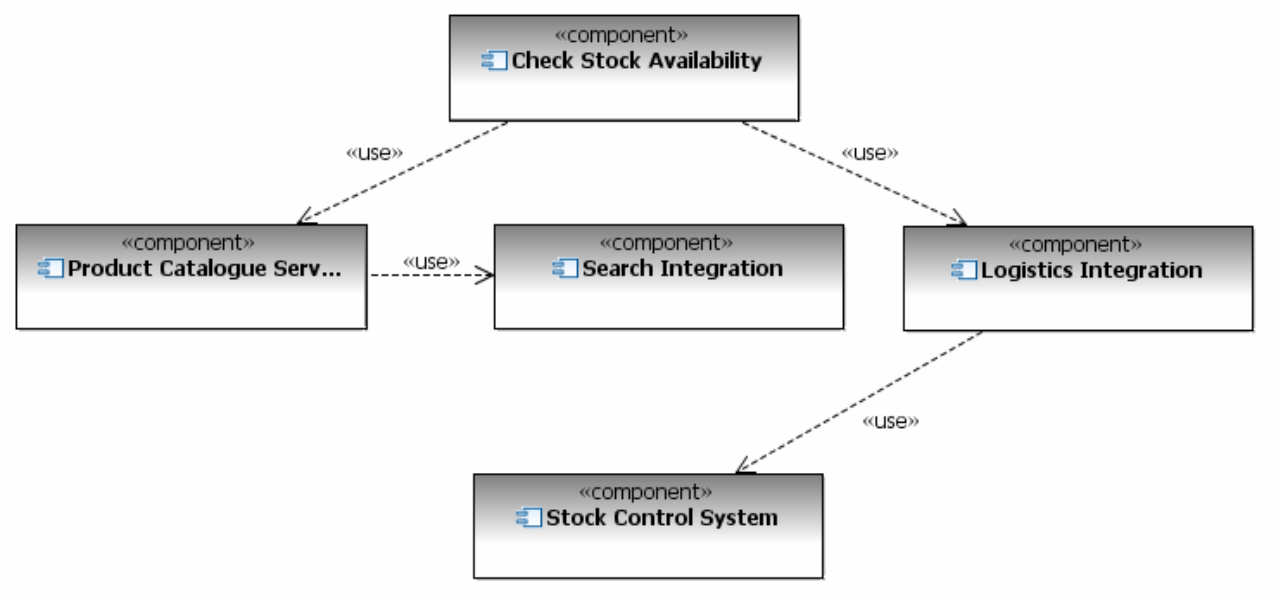


There is a separate use case description for each of the business use cases shown in Figure 3. This description will show the steps in the use case, together with alternative flows. The goal of the Check Stock Availability use case is for the customer to find out whether an item is available in her local LiUS store. The main flow (only) for that use case involves these steps:

1. The Actor requests a stock query.
2. The system prompts the Actor to select a store name and enter either an article or the product name or number.
3. The Actor selects a store name and enters an article or product name. If a product name is entered, the user may also be required to specify further selection (color, for example) for that product.
4. The system searches for the selected article or product in the named store and checks whether it is available or not.
5. The system indicates that the article or product is available.
6. The system returns the availability information for the article or product.
7. The use case ends successfully.

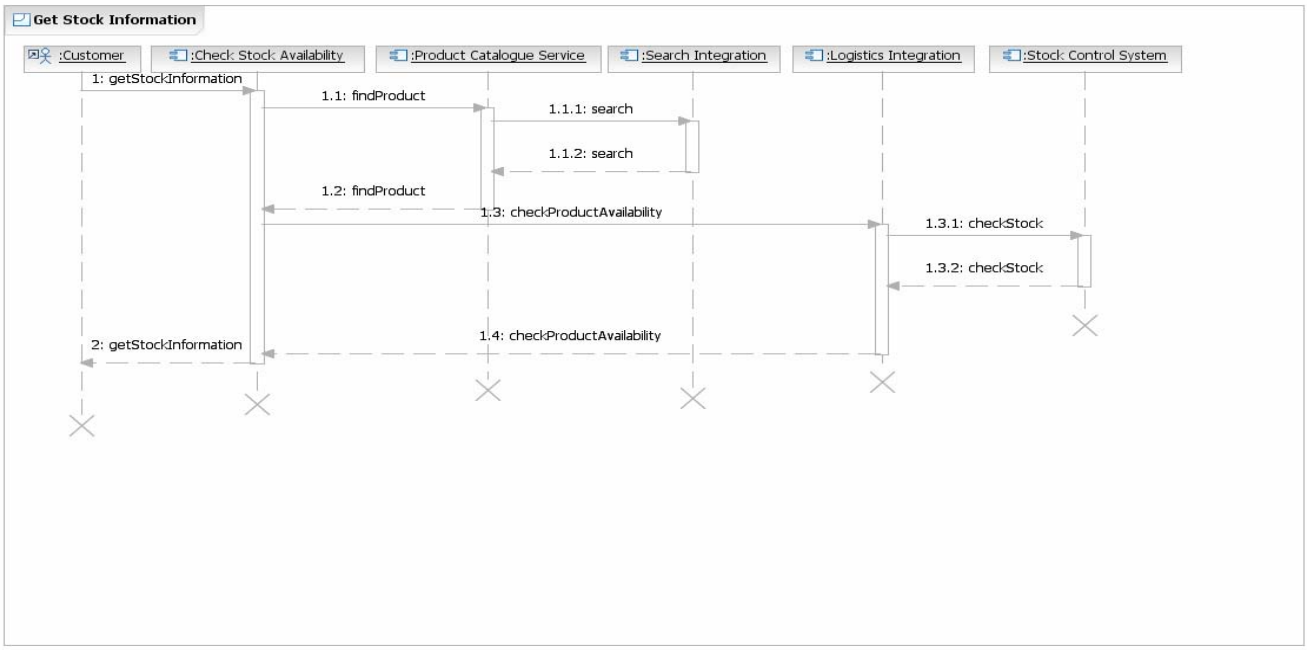
The application level of a Component model defines the system's application-level components (those that deliver the business requirements of the system as opposed to technical components, which deliver the application-independent behavior of the system), as well as their relationships and the way that they collaborate to satisfy the business requirements of the system. Figure 4 shows a partial application-level Component model for the LiUS system.

Figure 4. Partial Component model for the LiUS system



Collaborations between components that explain how they satisfy use cases are shown by using Unified Modeling Language (UML) sequence diagrams. Figure 5 shows a sample sequence diagram for the Check Stock Availability use case.

Figure 5. Sequence diagram for the Check Stock Availability use case



Technical cross-cutting viewpoint

This cross-cutting viewpoint is concerned with those components and their relationships that provide the application-independent behavior of the solution, meaning behavior that the system must provide to support the business purpose of the system. As with the Application cross-cutting viewpoint; this one is nearly always required. A business application must have technical infrastructure components that allow it to run.

Table 6 shows the impact that this cross-cutting viewpoint has on the basic viewpoints.

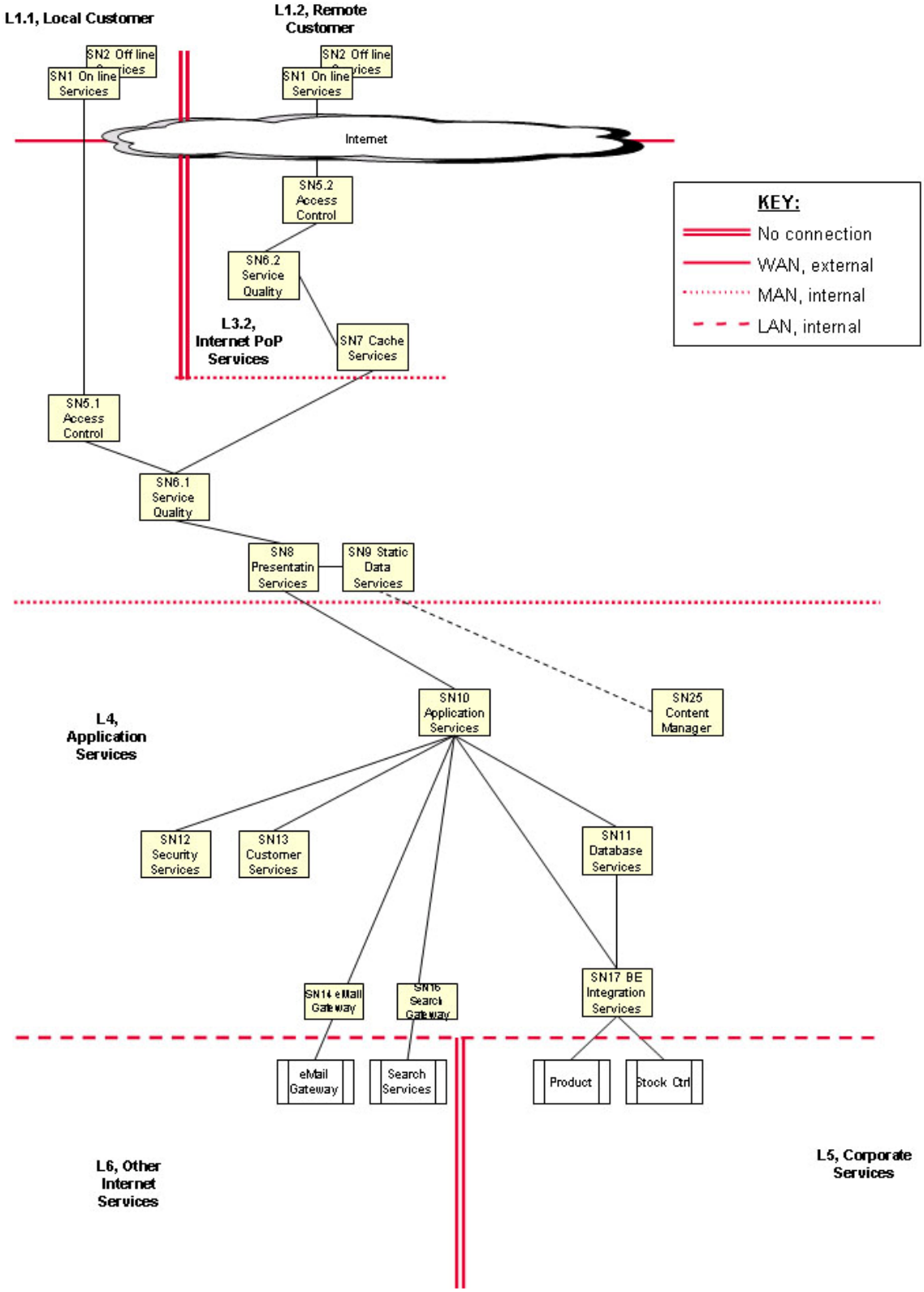
Table 6. Impact of the Technical cross-cutting viewpoint on the basic viewpoints

Basic viewpoint	Impact
Requirements	The Requirements view captures the requirements and constraints related to supporting the application-level functionality.
Functional	The Functional view is where the static and dynamic behaviors of the technical components of the system are captured. It includes component and data models that address the nonfunctional requirements of the system or that address application-independent requirements, such as intercomponent communication and persistence or transaction handling mechanisms.
Operational	The Operational view captures how the application-independent components and data are deployed on nodes and the required connectivity between them.
Validation	The Validation view is concerned with the validation of the technical constraints imposed on the solution.

Examples

The Operational model defines how the system's components get deployed to geographically distributed nodes and the connections between those nodes. Figure 6 shows the Operational model for the LiUS system, depicting the placement of technical components that support the business purpose of the system. Nodes are allocated to various locations (for example, L1.1 Local Customer, L4.1 Application Services, and so forth). Locations are separated by borders, which are solid or broken lines.

Figure 6. LiUS Operational model



Systems Management cross-cutting viewpoint

The Systems Management cross-cutting viewpoint focuses on the concerns of the systems management and operations people who must run and maintain the system when it has been deployed into production. Table 7 shows the impact that this cross-cutting viewpoint has on the basic viewpoints.

Table 7. Impact of the Systems Management viewpoint on the basic viewpoints

Basic viewpoint	Impact
Requirements	The Requirements view captures the system management requirements of the proposed solution. System management requirements may be functional in nature (for example, how a component should raise an exception to indicate business or technical failures), or they may be constraints that are placed on the system, such as whether there is a mandate that systems management software from a particular vendor must be used.
Functional	The Functional view includes guidelines for how to ensure that the right components and data structures are in place to support the system management requirements. Examples are components that provide event notification and logging or database tables that support error logs, and so on.
Operational	This view includes guidelines that show how to define nodes that support system management products and packages in the deployed system.
Validation	The Validation view describes how to verify that the system management functional requirements and constraints have been satisfied.

Availability cross-cutting viewpoint

The Availability cross-cutting viewpoint is concerned with the system's ability to be operational according to its specified availability requirements. It is also concerned with the system's ability to recover from failures that make the system fully or partially unavailable.

Table 8 shows the impact this cross-cutting viewpoint has on the Basic viewpoints.

Table 8. Impact of the Availability Cross Cutting viewpoint on the Basic viewpoints

Basic viewpoint	Impact
Requirements	The Requirements view ensures that the availability requirements of the proposed solution have been adequately stated. For example: "The system must be available to take orders between the hours of 0600 and 2200."
Functional	The Functional view includes guidelines that show how the system's components need to provide additional functionality that enable the availability requirements to be met. For example, when the system is unavailable because of network downtime, the users may need a method of working in an offline mode, and that may necessitate identifying additional components that will be required in such instances.
Operational	The Operational view will be significantly affected by availability concerns. It may need to accommodate automatic failover, clustering, or redundancy, for example.
Validation	The Validation view describes how to validate that the availability requirements have been satisfied.

Performance cross-cutting viewpoint

The concerns of the Performance cross cutting viewpoint focus on the ability of the system to meet its performance requirements (such as response time and throughput). Table 9 shows the impact this cross cutting viewpoint has on the basic viewpoints.

Table 9. Impact of the Performance cross-cutting viewpoint on the basic viewpoints

Basic viewpoint	Impact
Functional	The Functional view may indicate that some tradeoffs need to be considered, so that the system meets performance requirements. For example, a strict layering approach that has components in one layer communicating only with components in the same layer or the one directly below it may perform very poorly, thus resulting in a compromise that introduces a less-strict layering policy.
Operational	The Operational view is critical to the system's overall performance. This viewpoint includes performance models that show how the system will perform for a given set of inputs, plus Operational models that show how the system needs to be configured to support the performance requirements. This is usually decided by analyzing the results of executing the Performance model with different parameters.
Validation	The Validation view describes how to validate that the system performance requirements have been satisfied.

Examples

In this example. we consider what a (partial) Performance model for the LiUS system might look like and what information is needed to build it. A Performance model shows different performance results that could be achieved for different business processes by varying the fundamental units of work associated with components and nodes of the system. These fundamental units of work are referred to as the *atomic characteristics* of the system. They refer to the various units of work and their associated costs, which are

used when describing and measuring the qualities of an IT system. The Performance cross-cutting viewpoint would describe which atomic characteristics are important for building Performance models. In the case of LiUS system, these viewpoints could include the following characteristics:

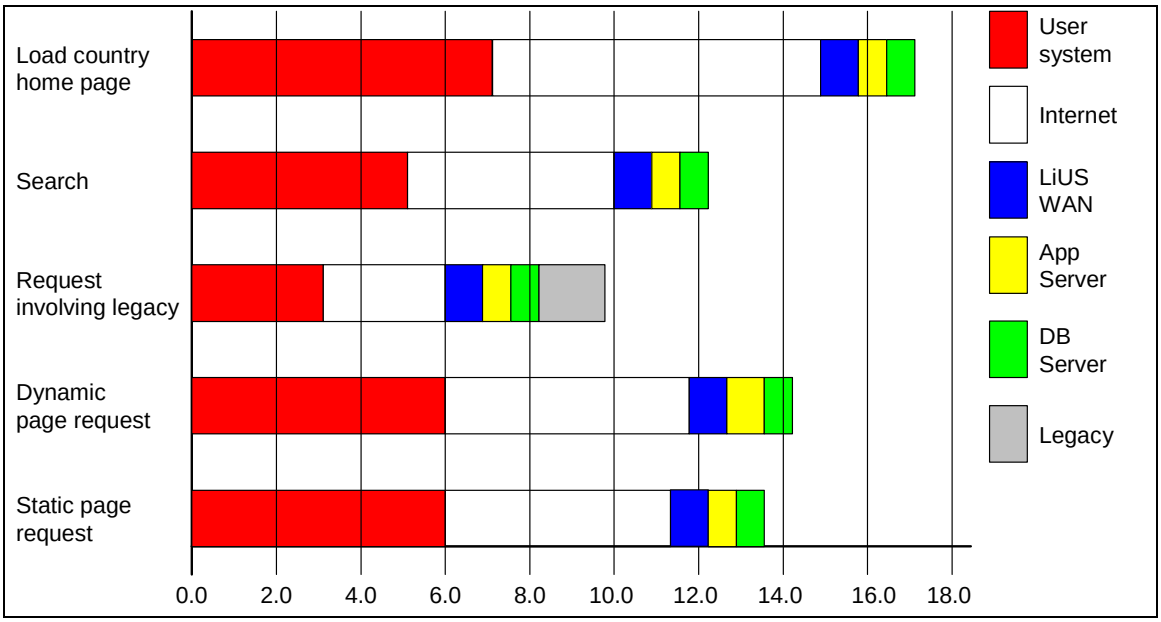
- Static and dynamic page requests per hour and per visit
- Requests involving legacy calls per hour and per visit
- Searches per hour and per visit
- HTTP request size
- HTTP response size

After gathering the atomic characteristics, the next step is to map the key business processes to the underlying technical components (nodes and connections, for instance) that support the execution of those processes. This is necessary to understand the impact of executing a business process on the underlying technical components that make up an IT system. If you know the atomic characteristics and understand how business processes map to components and nodes, it is possible to create a cross-cutting Performance view of the basic Operational view by building a Performance model of the system.

The simplest way to do this is to put the information into a spreadsheet that enables you to illustrate the results graphically. By feeding different configurations of the Operational model into the Performance model and adjusting the atomic characteristics as required, it is possible to perform different analyses of the performance of the system and look at the expected results of each.

Figure 7 shows the results from one run of the Performance model for the LiUS system. This particular model shows the overall response time for several different processes (load country home page, search, a request involving the legacy system, dynamic and static page requests), plus how that response time is allocated to different components and nodes in the system (user system, Internet, LIUS WAN, application server, database server, legacy system), based on the atomic characteristics exhibited by those nodes and a particular configuration of the system.

Figure 7. Performance model Extract for the LiUS System



Security cross-cutting viewpoint

The ability of the system to control who is authorized to access functions and resources is the primary concern of the Security viewpoint. Detecting and recovering from failures in the system's security mechanisms is also addressed by this viewpoint.

Table 10 shows the impact that this cross-cutting viewpoint has on the basic viewpoints.

Table 10. Impact of the Security cross-cutting viewpoint on the basic viewpoints

Basic viewpoint	Impact
Requirements	The Requirements view ensures that the security requirements of the proposed solution have been adequately stated. Security requirements may be functional in nature, such as how users authenticate themselves to the system, or they may be constraints that are placed on the system, such and what strength of encryption must be used when encrypting data in the system.
Functional	The Functional view ensures that the right components have been identified to implement the security requirements, for example components that handle authentication, authorization, auditing, nonrepudiation, and such. This viewpoint may also show which components and data elements need to be protected.
Operational	The Operational view will include guidelines that show how the system’s operational environment can be affected by security requirements, including the need for specialized hardware or software (for example, firewalls).
Validation	The Validation view describes how to validate that the systems security requirements and constraints have been satisfied.

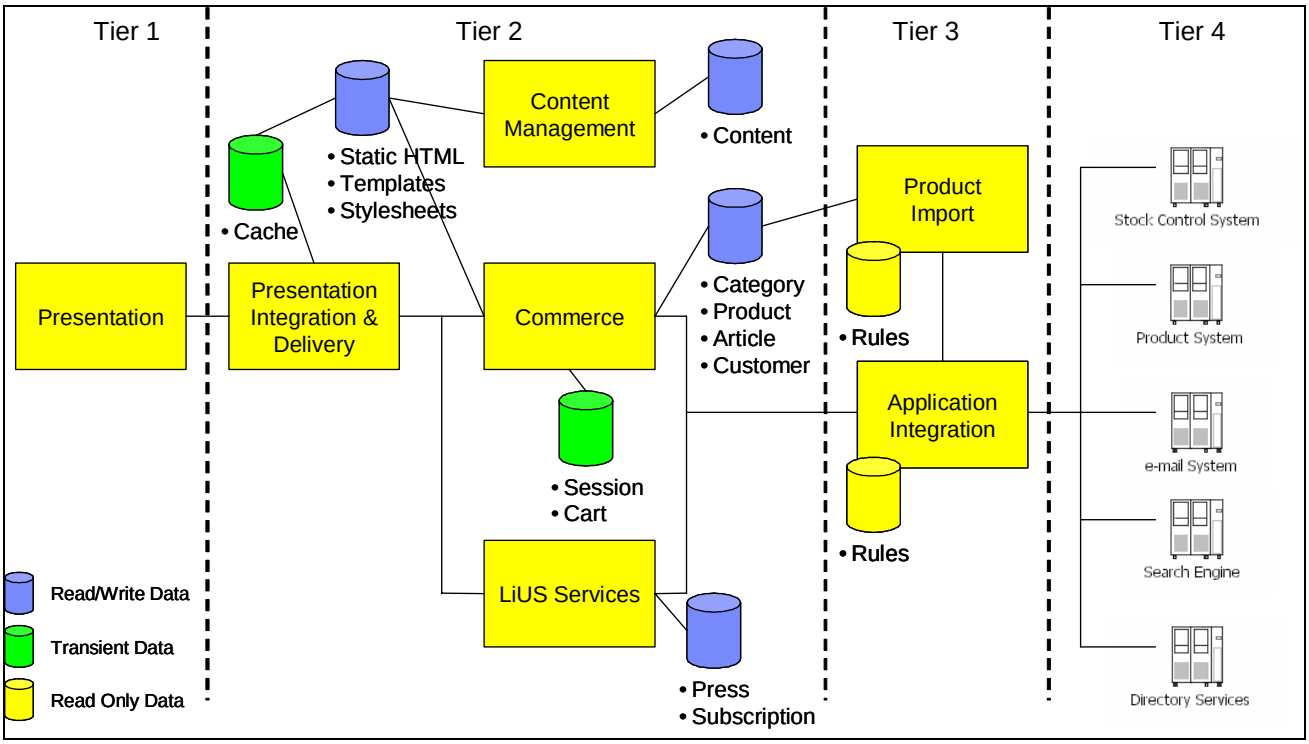
Other key viewpoints

Some viewpoints assemble elements from multiple cells across the viewpoint framework into a summary. Even though they could be organized by cell on the viewpoint framework, they typically are bundled into one common artifact.

Architectural Decisions are a good example of this type of view. These are important decisions about any aspect of the architecture, often documented in a single artifact that captures key decisions made, together with the options considered and the rationale for selecting a particular option.

Similarly, an Architecture Overview can be constructed that references elements in multiple cells across the viewpoint framework. Figure 8 gives an Architecture Overview for LiUS as an example. This diagram shows combined functional-application and operational-application views for LiUS.

Figure 8. Example of an architecture overview



Summary

The framework presented in this article is an evolution of Philippe Kruchten's 4+1 View Model of Software Architecture. IEEE 1471 advanced and standardized the concepts introduced by that model, thereby providing a foundation for the IBM framework of views and viewpoints. This framework provides an extensible way to organize the description of a system so that aspects of it can be thought through and understood by all of its stakeholders.

The IBM IT System Viewpoint library is a catalog of viewpoints built on that framework for the description of IT systems. Its basic and cross-cutting viewpoints specify different representations of an IT system to address stakeholder concerns.
By providing a standard way to organize a system description, as well as a set of conventions for notation, terminology, and semantics to be used in that description, the framework and the viewpoint library enable consistency for IT system description across IBM unified methods.

Acknowledgements

The authors thank the following Unified Method Framework team members who contributed to the ideas in this article:

- **From IBM Global Business Services:** Bruce Anderson, Ian Charters, and Nicholas Whidborne
- **From IBM Global Technology Services:** Adam Newth
- **From IBM Rational:** David L. Brown, Peter Eeles, Margaret Hedstrom, and Kelli Houston

Citations in this article:

- **IEEE1471:** IEEE Architecture Working Group, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Std 1471-2000, IEEE, 2000.
- **SPEM06:** OMG. Software Process Engineering Meta-Model (SPEM 2.0), OMG Submitted Specification ad/2006-08-01.
- **UML05:** OMG. Unified Modeling Language: Superstructure Version 2.0, OMG Formal Specification formal/05-07-04.
- Kruchten, Philippe. *Architectural Blueprints:* The "4+1" View Model of Software Architecture, IEEE Software 12 (6), November 1995.
- Rozanski, Nick and Woods, Eoin. *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*, Addison-Wesley, 2005.
- Spaas, Philippe. System Description Standard - Semantic Specification, R 3.0, September 2007.

Appendix: Mapping the 4+1 view model of software architecture to the IBM IT System Viewpoint library

This appendix describes how the 4+1 views map to the IBM IT System Viewpoint library. The IBM® Rational Unified Process® (RUP®) Version 2003 description of architecture is based on Philippe Kruchten's 4+1 views. Therefore, some mappings are described in terms of the RUP implementation of the 4+1 model.

Table 1. 4+1 to IT System viewpoint mapping

4+1 View	IBM IT system viewpoint
Logical	Functional
Process	Functional Operational Performance Availability
Physical	Operational
Development	Functional
Use Case	Requirements

Logical view

The Logical view describes the functional aspects of the system and defines its main subsystems and classes. These concerns are addressed by the static models of the Functional viewpoint.

Process view

The Process view captures the dynamic aspects of the system, and it is concerned with nonfunctional requirements, such as performance, scalability, availability, concurrency and distribution, and fault tolerance.

Many of these concerns are addressed by cross-cutting viewpoints in the IBM IT System Viewpoint library, but some are addressed solely by basic viewpoints. These distinctions are made according to the definitions of basic as opposed to cross-cutting views in the IBM® Views and Viewpoints Model of IT Systems framework.

The concurrency aspect of the Process view addresses the concerns of application architects who need to understand the processes and threads the system will use and the communication mechanisms used to coordinate their operation. The Functional viewpoint addresses the process interaction aspects of the Process view through interaction diagrams within a functional model. Components are mapped to concurrency elements to describe what parts of the system will run concurrently and what communication takes place to coordinate their execution. Although the Logical view was

focused primarily on the static description of a system, the Functional viewpoint also describes its dynamic aspects.

The performance and availability concerns of the Process view are addressed by cross-cutting viewpoints referenced in the main article that this appendix is intended to accompany. Scalability and fault-tolerance are likely candidates for additional cross-cutting viewpoints.

Physical view

The Physical view (called the Deployment view in RUP v2003) defines the hardware topology on which the system is executed. This view shows how the various runtime components are deployed and how they communicate.

The Operational viewpoint addresses the concerns of the Physical view and broadens its scope. It is also concerned with the necessary systems management functions and activities needed to maintain components (for example, software distribution, responding to alerts, and so forth).

Development view

The Development view (called the Implementation view in RUP v2003) focuses on the organization of the software in its development environment. The software is packaged into modules that can be developed by one or a small team of developers, and subsystems are organized into layers.

The Development view's concerns are addressed by the Functional viewpoint. In terms of artifacts, the packaging structure should be visible at the physical level (implementation-specific model) of the component model.

Use Case view

The Use Case view unifies the other four views into a cohesive solution. Key use cases (considered to be architecturally significant) are selected to illustrate how the elements of the architecture interact to provide the system's functions.

The Requirements viewpoint addresses the concerns of the Use Case view.