

IBM Mainframe Bits: Understanding Architecture

Keith Winnard

Rob Hunt

Jo Johnston



z Systems

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get personalized notifications of new content
- ▶ Link to the latest Redbooks blogs and videos

Get the latest version of the Redbooks Mobile App



Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK



IBM Mainframe Bits: Understanding Architecture

This IBM® Redpaper™ publication reviews the role of the architecture and how it forms part of the solution by meeting the needs that drive the requirements for computer systems. It also examines the architecture's relationships with hardware and software, and offers considerations to help understand how these relationships work to ensure longevity and compatibility.

This paper includes the following topics:

- ▶ Introduction
- ▶ Understanding what an architecture must achieve
- ▶ Architecture and components

This paper is the first in a new series of “Mainframe Bits” that are designed to provide you with succinct information about discrete topics that relate to IBM z Systems™ architecture.

Introduction

To grasp the concepts that are presented in this paper, a clear definition of *architecture* within the context of computing systems is essential. We use the following definition from *z/Architecture Principles of Operation*, SA22-7832-09¹:

The architecture of a system defines its attributes as seen by the programmer, that is, the conceptual structure and functional behavior of the machine, as distinct from the organization of the data flow, the logical design, the physical design, and the performance of any implementation. Several dissimilar machine implementations may conform to a single architecture. When the execution of a set of programs on different machine implementations produces the results as defined by a single architecture, the implementations are considered to be compatible for those programs.

Machine definition: For this publication, *machine* in the definition refers to a physical computer.

Given this definition of architecture, you might have the following questions:

- ▶ What tangible purpose does the architecture achieve?
- ▶ Why is the architecture important?
- ▶ How do you determine what lies within or outside of the scope of the defined architecture?

¹ *z/Architecture Principles of Operation*, SA22-7832-09:
<http://www.ibm.com/support/docview.wss?uid=pub1sa22783209>

- How does the architecture align with the IT infrastructure that supports the business?
- Is an architecture part of the machine?

In this paper, an *architecture* is considered a conceptual representation of or framework for a solution. The remainder of this paper helps you understand this concept and reviews what implications and considerations are required to put an architecture in place.

Understanding what an architecture must achieve

The architecture is the part of a solution that is created in response to a collection of needs. These needs can come from the following sources:

- Consumers and customers
- Line of business, including users
- Business Partners or vendors
- Government, legal, and regulatory requirements
- Financial or business plans and strategy
- Innovations in products or technology
- Industry (for example, healthcare and education)

The needs can be from any of these sources, a combination of the sources, or a different source. A solution must be put in place to meet these needs. In essence, the needs act as the driver for creating the solution.

Needs and solutions

We start with a collection of needs and define a solution to meet those needs (see Figure 1).

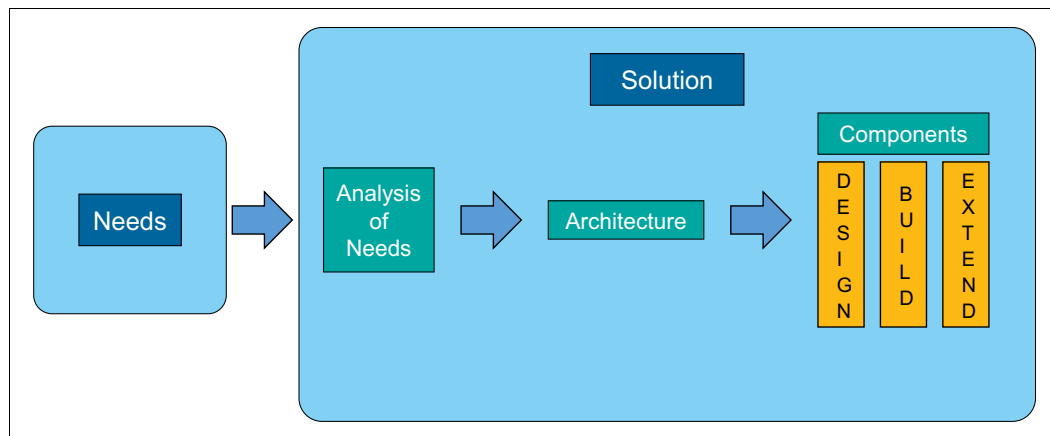


Figure 1 Needs and solution simple flow

Figure 1 shows a simple process that moves from left to right. The development of a solution consists of the following main stages:

- Analysis of needs
- Design of the architecture
- Build components that are based on the architecture

An architecture defines a conceptual structure for components to be built by using a framework. The solution is intended to meet the identified needs and is acceptable if the needs do not change beyond the scope of the solution.

But what if the needs change? If new or different needs emerge, how is the solution affected? In this case, the analysis of the “new or changed” needs is the same, the architecture must be reviewed to see whether changes are necessary. New components or enhancements to components might be necessary to provide the functions to meet these changed needs.

Given a change in needs, is a new architecture necessary? If the architecture can accommodate the new requirements, only changes to the components are necessary. Changes do not necessarily mean changes to the architecture, just to the components. Not every component uses all of the architecture to satisfy a need (see Figure 2).

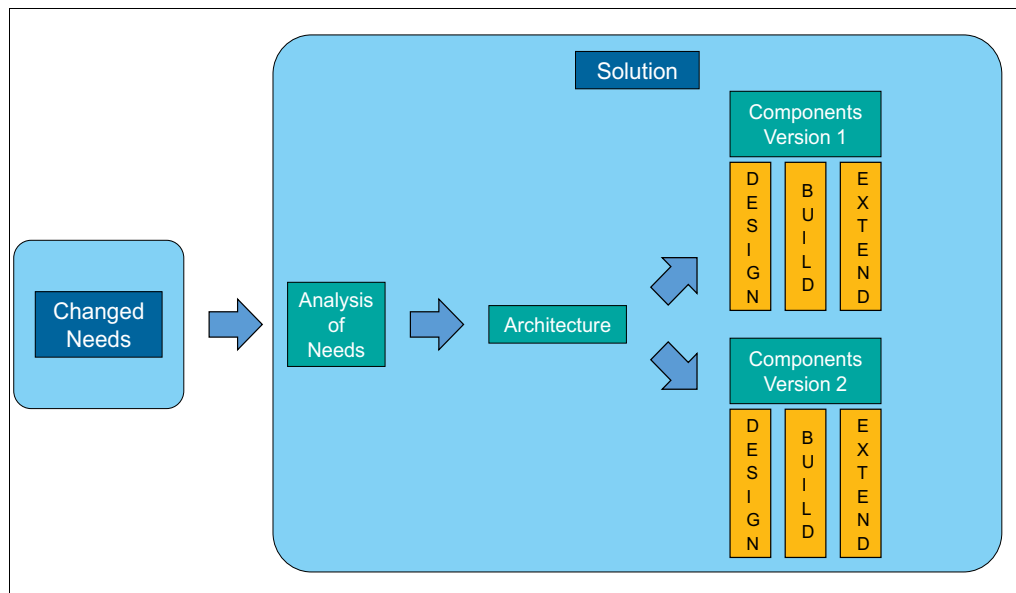


Figure 2 Same architecture with multiple components

If you considered the components that are shown in Figure 2 as computers, you can see that the following computers are used:

- Version 1 was the original solution to the original needs.
- Version 2 was developed because the original needs changed and new functions was required to provide a solution to the changed needs.

Before going further, consider an important implication: If the architecture is to remain true to its purpose, what is the relationship between the different versions of components that uses the same architecture? Compatibility is a crucial consideration for an architecture to be successful and achieve longevity.

Compatibility

Computer Version 1 satisfied the original needs, which are referred to as *Needs Level 1*. Changes to existing needs or new needs are referred to as *Needs Level 2*. However, Version 1 computer cannot satisfy Needs Level 2. It can meet Needs Level 1 only. To meet Needs Level 2, the functions of the computer must expand and change. These needs cause the creation of Computer Version 2 that meets the needs of Needs Level 2.

So far so good, but there might be a problem. Will Computer Version 2 meet the needs of Needs Level 1? If Version 2 remains true and within the framework of the architecture, it will meet the Needs of Level 1. If it was not built to the framework of the architecture, it might not satisfy Needs Level 1.

The ability of Version 2 to support both Needs Levels is known as *compatibility*. To be more specific, this kind of compatibility is known as *backward compatibility*. Backward compatibility implies that if a new version of the computer is designed and built, it can process the requests of the latest needs and previous needs levels.

Backward compatibility can be achieved only if the architectural framework allows it.

Figure 3 show the compatibility between the Needs Levels and the Computer Versions.

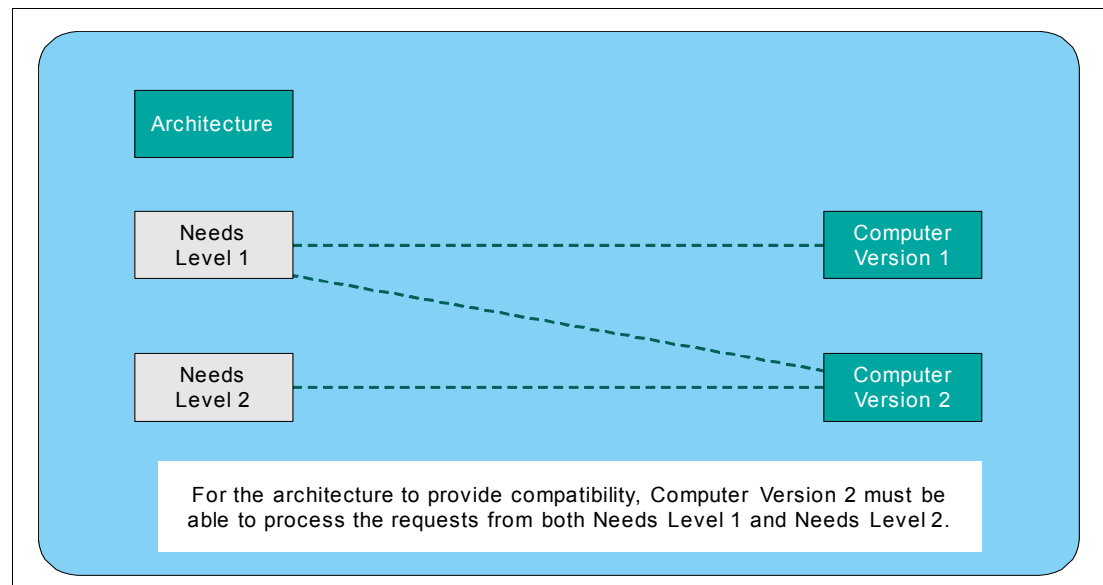


Figure 3 Compatibility within the architecture

The dashed lines in Figure 3 identify the relationship between the Needs Levels and the Computer Versions. Consider the following points:

- ▶ Computer Version 1 and Version 2 are both within the same architecture.
- ▶ Computer Version 1 can process requests from Needs Level 1, but not requests from Needs Level 2.
- ▶ Computer Version 2 was built with extended features that can process requests from both Needs Levels.

Why is this issue significant? Expanding the architecture protects the organizations that purchase the computers from the following exposures:

- ▶ Having to buy more computers each time needs levels change or increase.
- ▶ If there are modified Needs Levels, the previous computer might become redundant and the organization wasted resources and investment on the previous version.
- ▶ The data and information that is used by each Needs Level might be interdependent, which causes more complications and possible restrictions if Computer Versions 1 and 2 are not compatible.

Instead of buying more computers, it is simpler for the organization to upgrade the existing computer to the next version to meet the new Needs Levels, as shown in Figure 4.

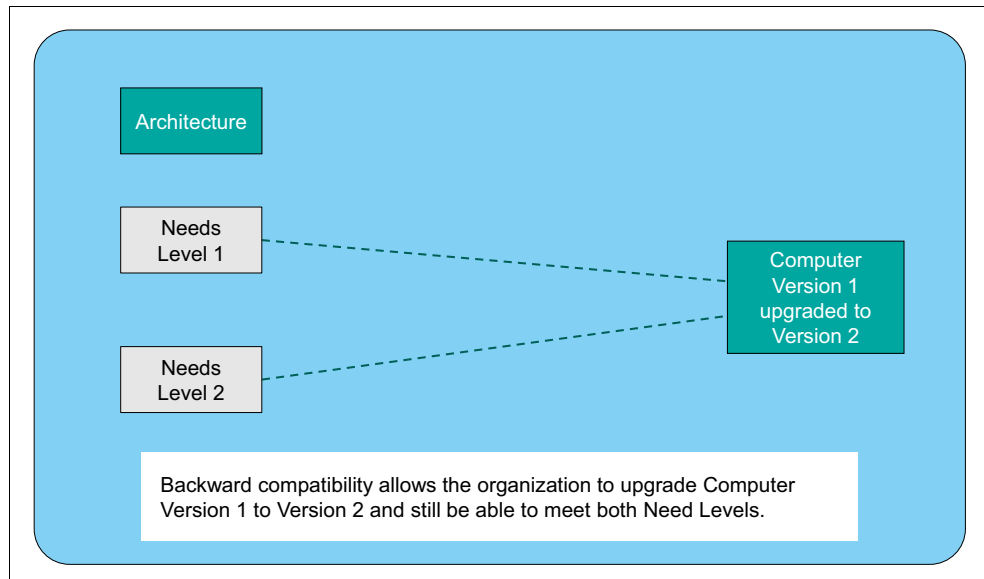


Figure 4 Backward Compatibility within the architecture

As the organization develops and consumer patterns change, new Need Levels can continue to arrive and some existing Needs Levels might be modified. It is likely that the relationships between Needs Levels can become more complex. This situation is why the design of the architecture is important. Consider the scenario that is shown in Figure 5.

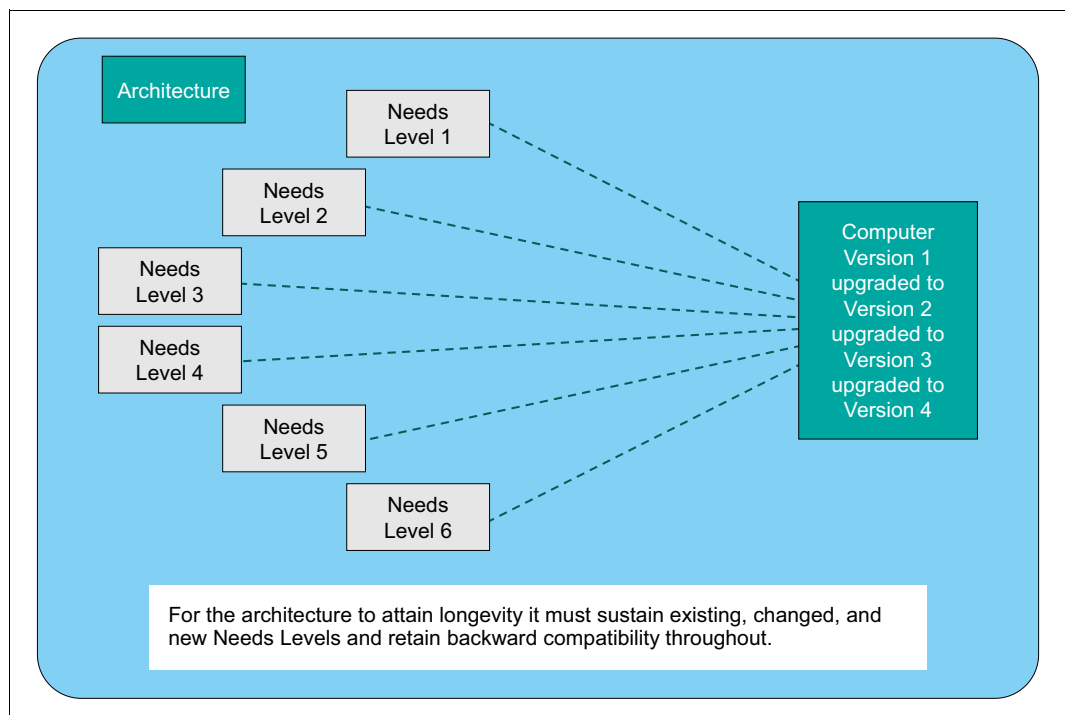


Figure 5 Architectural longevity

A new version of the computer might not be required for each new or changed Needs Level; therefore, the organization might not need to upgrade its computer for every change. The architecture enabled the computer to accommodate all the Needs Levels by offering the option to upgrade.

Managing larger scale Needs Levels

Organizations today often have more than one computer. This situation might be because the volume of activities that are undertaken are beyond the capacity of a single computer even though it is not beyond the computer's capabilities. There are likely other reasons, but those reasons are outside of the scope of this paper.

The use of multiple computers to meet Needs Levels is another major consideration for the architectural design. Not only must computers process the requests from the Needs Levels, they must manage these activities as a group and communicate with one another. Figure 6 shows how the use of multiple computers that are grouped is a viable solution for an organization with many Needs Levels.

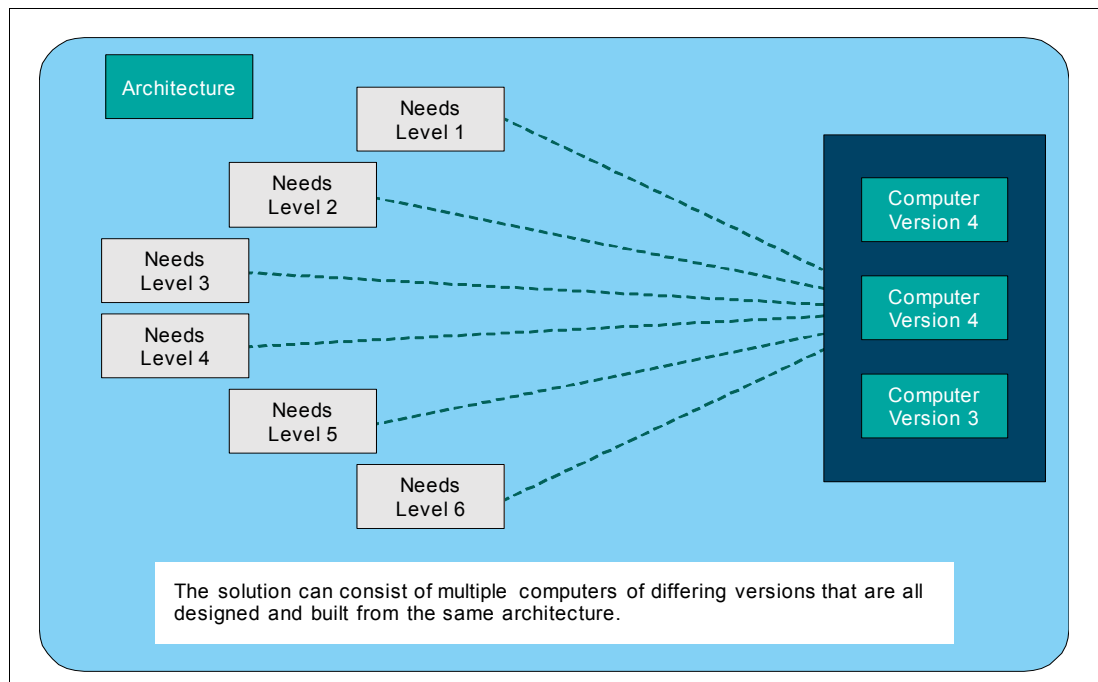


Figure 6 Multiple computers within the same architecture

The increase in Needs Levels places higher demands on the computers and for large organizations, this increase might be beyond the capacity of a single computer. The ability for computers to work together to process high volumes of requests is imperative for large organizations. The computers can transfer work to and from each other or share process activities from the same Needs Levels. If one computer supports only lower Needs Levels, any higher Needs Level requests must be directed to a computer within the group that can support these higher Needs Level requests.

Thus far, we described how the computers can be extended to add new functions. The following issues beyond functions can cause computers to be extended:

- ▶ **Speed**

The computer has the functions, but needs a faster level of performance.

- ▶ **Availability**

The organization needs the services to be available all the time so the computer must be highly reliable. Therefore, it must recover from component failures or seamlessly switch to an alternative component if a failure occurs.

- ▶ **Security**

The organization might be subject to requirements that stipulate that certain aspects of its data must be encrypted or perhaps access to data is restricted.

There are other considerations, but these reasons show why the computers might expand their functions to improve their capabilities and value to the organization.

Outgrowing the architecture

What happens when the Needs Levels exceed the capabilities of the computer and the architecture? This situation can mean that the computer can no longer be expanded within the architectural framework. Therefore, the architecture cannot provide a solution to satisfy the Needs Levels.

Consider this serious issue from the perspective of the organization that uses the architecture and solutions to meet its needs. The effect on the organization, associated organizations, customers, partners, business applications, and workforce can be far reaching. The fabric of the organization's ability to exist might be put at risk. This problem can result in the following suggested responses:

- ▶ Move to another platform architecture
- ▶ Modify the existing platform architecture

The first suggestion involves some form of conversion or migration to another set of solutions that are based on a new architecture that meets the new Needs Levels. This option demands careful consideration and planning. An organization often faces the following typical challenges:

- ▶ New business applications
- ▶ Archived data and having programs to access that data
- ▶ Skills within the workforce
- ▶ Conversion or migration effort
- ▶ Conversion or migration costs
- ▶ Risk to the organization
- ▶ Backward compatibility
- ▶ Security exposures
- ▶ Legal requirements
- ▶ Expected longevity of the target architecture
- ▶ Change freeze on applications during the conversion or migration
- ▶ Compatibility of the new solutions with partnerships
- ▶ IT Infrastructure changes (hardware, software, processes, procedures, and support)

There are other challenges and their range and intensity vary from organization to organization.

The second suggestion is aimed at the architecture supplier. The supplier must continually evaluate the needs of the consumers and forecast needs and protect the consumers' needs to avoid a situation whereby the architecture becomes obsolete and poses a threat to the consumers.

As you saw components' functions expanded to meet new Needs Levels, the architecture must expand and accommodate more functions for the components that form the solutions offered to organizations. The supplier often faces the following typical challenges in extending an architecture include:

- ▶ Backward compatibility for solutions
- ▶ Extend the architecture rather than replace it
- ▶ Avoid functional redundancy
- ▶ Expand with further growth in mind
- ▶ Prevent the need for complicated migrations
- ▶ Provide for component upgrades
- ▶ Continue to meet the consumers' needs
- ▶ Position to meet projected consumers' needs

The key point is that the architecture should *add to* its capabilities and not *replace* capabilities to avoid issues with backward compatibility. Without the backward compatibility, you might assume that the architecture was replaced rather than expanded.

Figure 7 shows how the architecture and component expansion meet the organizations' needs.

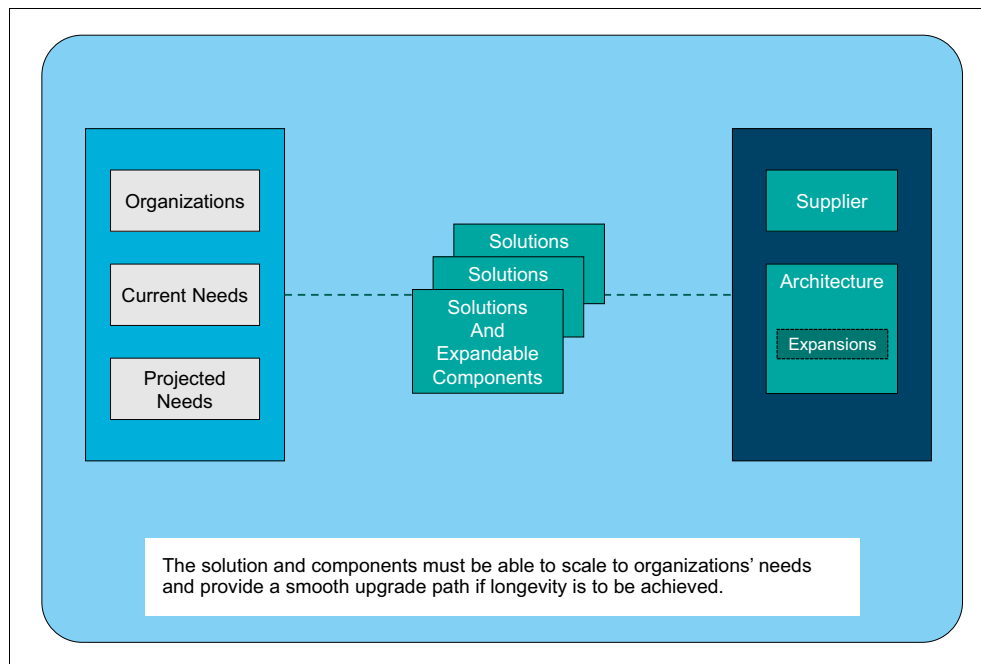


Figure 7 Architecture expansion

The architecture achieves longevity if it can satisfy current and future needs and remain flexible. In addition, the components must continue to expand in alignment with the architecture to minimize disruption and provide organizations with an upgrade path to meet all their envisaged needs.

Architecture and components

The definition of architecture (as described in “Introduction” on page 1) states: “The architecture of a system defines its attributes as seen by the programmer, that is, the conceptual structure and functional behavior of the machine...” and then continues by distinguishing the architecture apart from other areas by stating “...as distinct from the organization of the data flow, the logical design, the physical design, and the performance of any implementation.”

Based on this definition, you can surmise that the conceptual structure and functional behavior of the machine as seen by the programmer is our guideline to understanding more about the architecture and what it does or does not entail. The conceptual structure provides programmers with a solid base to design solutions knowing that while they remain within the conceptual structures, the solution works because they understand the functional behavior of the machine.

The need for scaling up is covered by the remainder of the definition, which states: “Several dissimilar machine implementations may conform to a single architecture. When the execution of a set of programs on different machine implementations produces the results as defined by a single architecture, the implementations are considered to be compatible for those programs.”

A key point here is that the architecture governs the components of the computer. Although the components might change logically and physically, they remain compatible and functional if they conform to the architecture. If the components drive the architecture, the integrity of the architecture is challenged and can lead to architectural incompatibilities and hence failure.

Looking at components and other layers that perform different roles, all of them play a part in providing an organization with a solution. These combinations help to further clarify the role of the architecture and its relationship with all the components that offer a solution.

Consider the following areas:

- Compute

The ability for the machine to perform the appropriate action on data that is presented to it is essential. Actions, such as a calculation, copy, or comparison, are typical. The compute function also contains memory, which allows for the data to travel to and from the areas where the compute functions can be performed. However, be aware that there are different levels of memory.

- Storage

Programs must be stored for retrieval and execution. Data also must be stored when not in use.

- Network

The components must communicate with each other on the same machine, between other machines of the same architecture, and to networks with many different devices, such as different computers and personal digital devices via the Internet.

If you take these three areas (see Figure 8 on page 10), plan how each area should work in terms of a conceptual structure and functional behavior, and define those areas as the base allowing its functions to be expanded and maintained, you have a strong architecture on which components can be built to service the business needs.

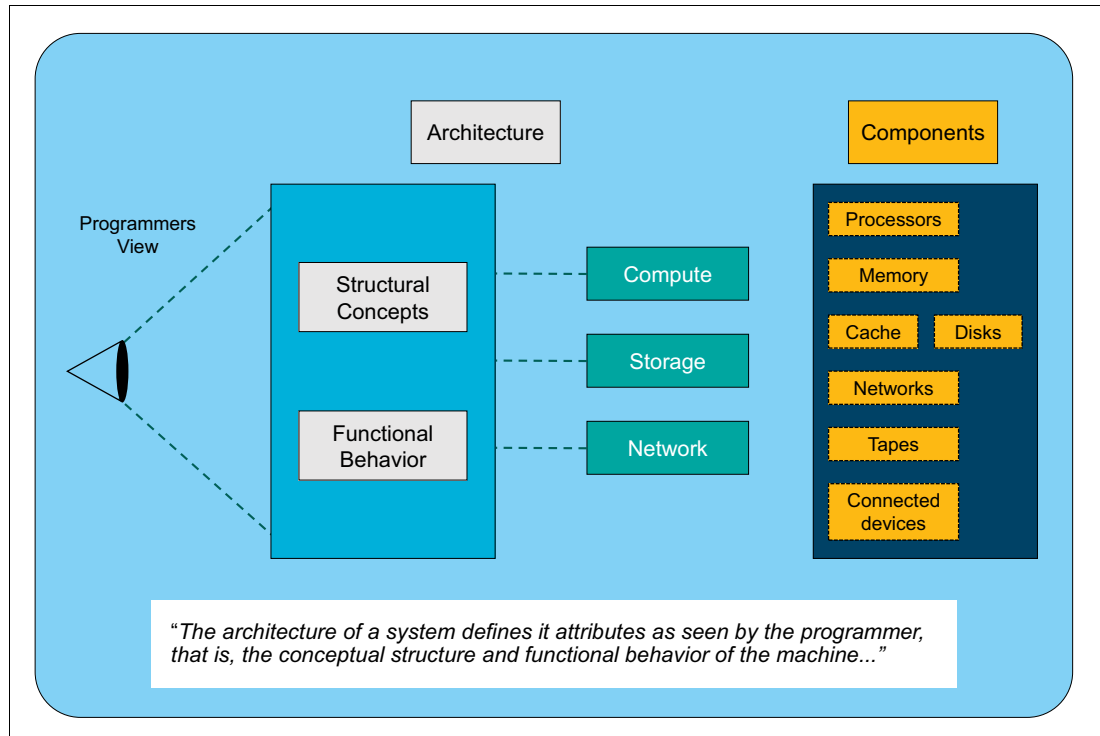


Figure 8 Programmer view of architecture

The consistent view for the programmer is the architecture. The programmer might have an awareness of the components, but these components can change and the programmer might not be aware of the changes. Consistency and compatibility are maintained if the programmer can understand the architecture.

The components relate to the three base areas of the architecture and can change or expand. If the architecture is adhered to, the changes or expansion are acceptable. There might be more components; the components that are shown in the Figure 8 are a simple base to show the relationship with the architecture.

Instruction set

Thus far, we described the organizations' needs and solutions. The solutions are comprised mainly of an architecture with components that were designed, built, and expanded to suit new needs. We also described the base areas of the architecture; specifically, the components that relate to the hardware of the computer.

The next area of consideration is the instruction set. The instruction set can be regarded as the *language of the machine*. While not an official definition, it helps clarify where the instruction set fits in to the overall picture. The instruction set is defined based on the architecture and then built as part of the hardware so that physical components can provide the expected functional behavior of the machine. Microcode also is added to assist with running functions within the hardware.

The next section describes other components that establish different layers that are built within the scope of the architecture. All of these layers provide organizations with solutions to meet their needs. The hardware aspects that relate to the architecture are shown in Figure 9 on page 11.

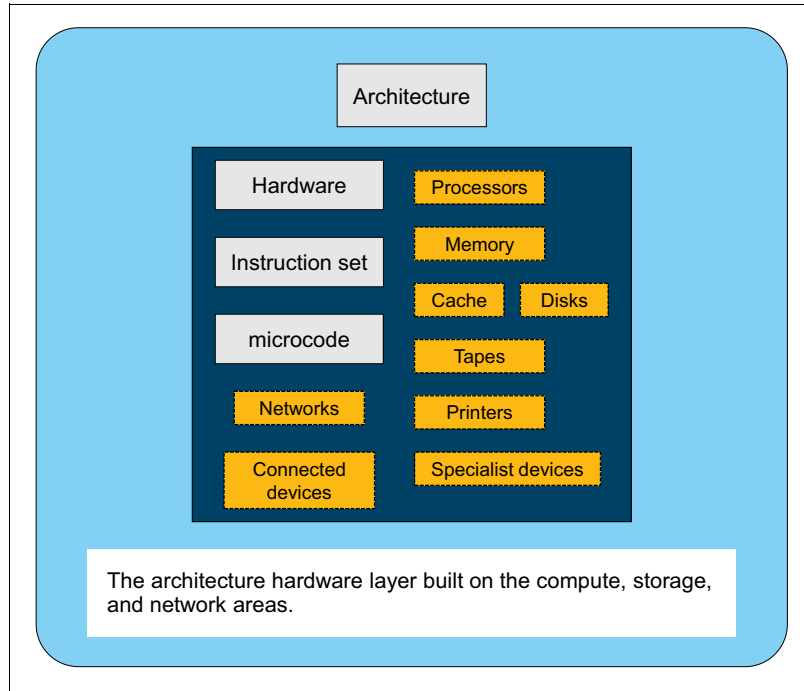


Figure 9 Architecture hardware layers

Operating systems and software

The next layer is the operating system. An operating system provides the next base level for business applications, user programs, specialist security software, and other software components, such as online transaction handlers, and systems management software to run. Figure 10 shows the software components of the system.

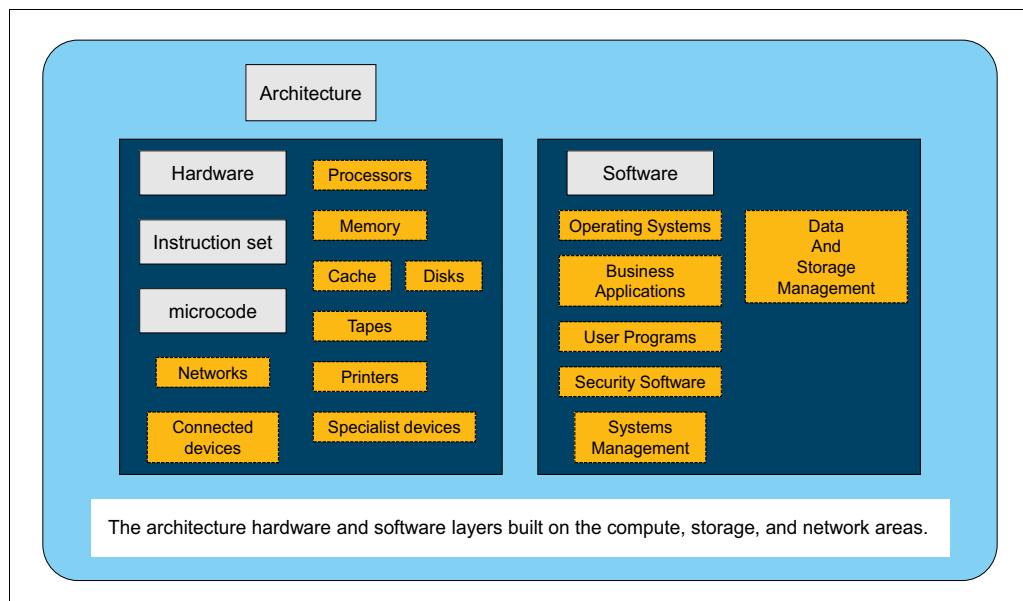


Figure 10 Architecture software layers

Figure 10 also shows the basic components that bring the architecture from a concept into a reality.

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Keith Winnard is the IBM Redbooks® Publications Project Leader for IBM z/OS® and related topics at the International Technical Support Organization, Poughkeepsie Center. He joined IT in 1977 and has worked for various clients and Business Partners. He is experienced in blending traditional z/OS environments and applications with web middleware and applications, and has presented on many mainframe-related topics.

Rob Hunt is an Accredited IT Specialist with the IBM System z® GTS Strategic Outsourcing Team in the United Kingdom. Rob has over 27 years of experience with IBM in storage, security, and systems management, supporting IBM MVS™, IBM z/VM®, and IBM z/OS environments. He has provided IBM mainframe storage and virtual machine support in the government, financial, retail, and insurance sectors.

Jo Johnston is a Certified IT Specialist and Chartered Engineer, who works in the IBM System z Strategic Outsourcing Team in the United Kingdom. She has worked on IBM mainframe systems as a systems programmer supporting z/VM, z/OS, MVS, IBM CICS®, IBM DB2®, IBM WebSphere® Application Server, and IBM IMS™ for more than 30 years.

Thanks to LindaMay Patterson of the International Technical Support Organization, Rochester Center, for her contributions to this project.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®	MVS™	WebSphere®
DB2®	Redbooks®	z/OS®
IBM®	Redpaper™	z/VM®
IBM z Systems™	Redbooks (logo)  ®	
IMS™	System z®	

The following terms are trademarks of other companies:

is a trademark or registered trademark of Ustream, Inc., an IBM Company.

Other company, product, or service names may be trademarks or service marks of others.



REDP-5345-00

ISBN 0738455180

Printed in U.S.A.

Get connected

