

# Preface

---

A new era of opportunity is dawning for Enterprise IT. Recent developments in the IT industry point to the exact antithesis of the perfect storm condition that led to the dot-com crash in the late 1990s. In the speculative environment of that era countless new players entered the marketplace. However, in most cases there was nearly total divergence between the technologies proposed by these new players and the ROI, or even business value to the intended audience—if an audience was defined at all.

On the business side there has been considerable consolidation and soul searching after the dot-com crash where the marketplace has essentially weeded out most nonviable ideas.

It is not easy as saying that the brick and mortar companies “won” because they had more “substance.” The economic landscape at the beginning of the third millennium is quite different than the climate at the start of the dot-com revolution, with new players, and old players gone or greatly diminished.

What strategies can be applied to optimize business outcomes in this business climate? In the next few chapters we will take a look at the confluence of the three technologies in the title of the book, namely, virtualization, service orientation, and computing grids.

## **A Bit of History**

It would be wonderful news if we could claim that virtualization, service orientation, and grids represent the latest and greatest in terms of emerging capabilities. Alas, we can’t lay that claim. The best we can claim is that the trio is old wine in new bottles. For instance, let’s look at

virtualization. The first efforts at virtualization can be traced back at least five decades to virtual memory and machine virtualization research done at IBM in the early 1960s for the IBM 360/67<sup>1</sup>, the demand paging research for the Atlas computer at the University of Manchester<sup>2</sup>, and the segmented memory design of the Burroughs B5000<sup>3</sup>, to list just a few of many examples. If we include assemblers and compilers as a form of virtualization, we can push the clock back by at least another decade.

Compilers relieved computer programmers from the drudgery of using wire patches or writing programs using ones and zeros. A compiler presents an idealized or should we say *virtualized*, logical view of a computer. For instance, real-life computers have architectural oddities imposed by limitations in the hardware design, such as memory regions that cannot be used, or registers that can be used for only certain purposes. A compiler presents a uniform view of memory to the programmer. The programmer does not need to worry about details of computer architecture such as whether the machine is stack or register oriented, details that may be irrelevant to the application under development.

Likewise, service orientation has a similar vintage. The origins of service orientation can be traced at least to the early 1960s with the development of the Simula language at the Norwegian Computing Center in Oslo. Simula started as a simulation programming language, but also brought to light concepts that proved useful for general programming: it facilitated interoperability and re-use, reducing the cost and time to completion of software projects. Simula and derivatives found an application in VLSI design at Intel, Cal Tech, and Stanford, became the development platform for the influential Smalltalk language at XEROX PARC in the 1970s, and key concepts were incorporated in the C++ language still in use today.

Finally, opinions about the origins of grid computing vary widely. It is possible to argue that grids started in the late 1960s with the first efforts at networked computing, or in the early 1970s with the standardization of networking around Ethernet. Grid research as a method for collaborative, federated computing picked up steam in the late 1980s and early 1990s with projects on algorithmic sharing aimed at cycle harvesting such as Condor in the University of Wisconsin and Zilla at

---

<sup>1</sup> [http://en.wikipedia.org/wiki/IBM\\_CP-40](http://en.wikipedia.org/wiki/IBM_CP-40)

<sup>2</sup> J. Fotheringham, *Dynamic Storage Allocation in the Atlas Computer*, Communications of the ACM Vol. 4, No. 10 (Oct. 1961), pp. 435-436.

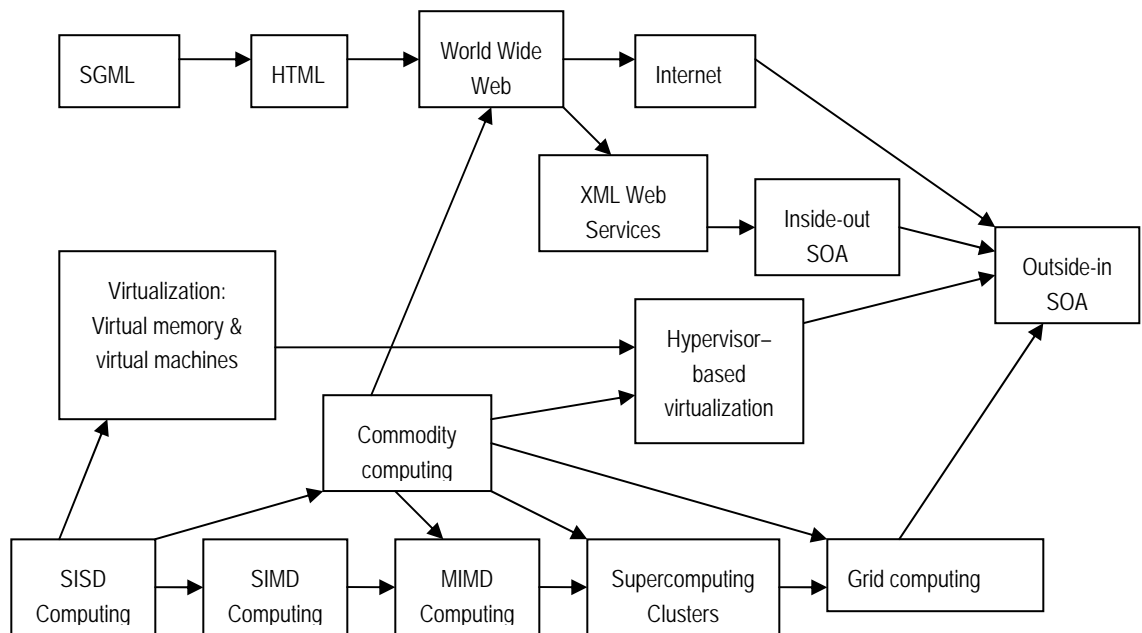
<sup>3</sup> [http://en.wikipedia.org/wiki/Burroughs\\_large\\_systems](http://en.wikipedia.org/wiki/Burroughs_large_systems)

NeXT Computer. A number of integration efforts ensued, such as the NASA Information Power Grid (IPG). Grid computing acquired a distinct identity thanks to the effort of the team led by Ian Foster at the University of Chicago.

The technologies for virtualization, service orientation, and grid computing have been amply documented in books and the research and trade literature. We do not attempt to go deeper or duplicate the excellent work that other authors have done.

Instead, we explore how the interplay between virtualization, service orientation, and grids are fundamentally changing the value economics of how information technology is delivered, and in the process, how organizations that depend on information technology to carry their day-to-day business are affected in turn. The authors believe we are witnessing a true inflection point.

Without an attempt to be exhaustive, Figure 0.1 summarizes the historical relationship between key technologies that eventually led to the concept of virtual service oriented grids.



**Figure 0.1** Technology Pedigrees for Virtualization, Service Orientation and Grids

The upper blocks start with the archetype Standard Generalized Markup Language (SGML) developed in the 1960s by Charles Goldfarb, Edward Mosher and Raymond Lone at IBM. The Hypertext Markup Language (HTML), a derivative of SGML, became the language of the Web and the Internet. The World Wide Web was initially used as a presentation interface for humans and certain aspects of it evolved into XML Web services for interoperable machine to machine communication. This machine to machine interface enabled modular composite service oriented architecture (SOA) applications, first within large enterprises (the inside-out model) and across enterprises small and large (the outside-in model). Chapter 2 covers these inside-out and outside-in models.

The middle blocks track the evolution of virtualization, initially applied to mainframes and eventually to computers based on commodity processors with hypervisors running as an intermediate layer between the hardware and the operating system.

The bottom blocks track the evolution of computer hardware, first with the single-instruction, single-data (SISD) style of computation initiated by mainframes in the 1950s. To improve throughput, certain computers were architected to apply a program to multiple data streams (single-instruction, multiple-data or SIMD.) These computers required data with a highly regular structure to take advantage of the extra power, and hence their applicability was limited. These restrictions were relaxed with SIMD computers, which allowed the constituent computers to operate on different data. Initially nodes in SIMD computers were linked together using proprietary high-speed interconnects forming supercomputing clusters all located in a single room. This setup was cost prohibitive for most applications except those requiring the highest performance. These restrictions were relaxed for grid computing where nodes can be geographically distributed, sometimes connected through the Internet.

The use of virtual service oriented grids is not yet pervasive in the industry. However, in this book we find quite a few examples, more often than not under different names. Organizations that use virtual service oriented grids will have a competitive advantage over those that don't and will likely have an economic influence disproportionate to their size.

This analysis would be of little practical value if it were not actionable. Hence, after we have established the dynamics of the interplay between the three undercurrents, we describe strategies and approaches that others have found useful. Using this insight, we attempt

to discover and document approaches that will likely be successful in this new environment.

We assume that the reader is familiar with the fundamental elements of virtualization, service orientation, and grids and has been asked to evaluate and even forge strategies for adoption in his or her organization or is seeking enough insight into the processes to be able to issue meaningful directives to staffers to have these strategies adopted in an organization.

Resources deployed under this new old environment are said to be deployed in a *virtual service oriented grid* or VSG. While these resources are best characterized by the qualities brought by each technology, the resources have evolved and they are not quite identical to their ancestor technologies.

- These resources are inherently distributed, federated and replicated as in grids seen in high performance computing applications, yet they feature more general capabilities.
- The service oriented nature of these resources makes them highly interoperable and logically they are closely aligned with business entities.
- The virtualization capabilities at play allow these resources to be logically “detached” from the physical hosts on which they run. This detachment can be seen in multiple ways: A powerful host may be capable of supporting multiple instances of a logical entity, and such a logical entity may be bound to its host only temporarily. The entity may migrate to other hosts for reasons of fault tolerance, disaster recovery or performance.

The presence of virtual service oriented grids does not bring fundamentally different capabilities. After all they run on the same computers, networks, and data centers as the more traditional information technology architectures.

Borrowing the term from requirements engineering, what virtual service oriented grids bring into the equation are *nonfunctional* capabilities. To contrast *functional* capabilities with nonfunctional capabilities, a change in functional capabilities changes the system behavior or function. This is not to say that changes brought up by virtual service oriented grids are not significant:

- A virtual service oriented grid environment will still be useful for processing home loans or performing credit card transactions.

x ■ The Business Value of Service Oriented Grids

- The performance of these systems will be infinitely scalable for practical purposes: if the workload slows the system, there will be a way of adding more resources to bring response times back in line without hitting a wall.
- New services can be put together by composing more primitive services using compatible protocols. If these primitive services are already available, the new services can be assembled almost automatically and in near real time.
- The component services can be *insourced* or *outsourced*. The size or *granularity* of the component services is much smaller than a traditional outsourced application such as payroll. The negotiation or handshaking to bring an outsourced component is automatic and machine driven as opposed to the months of negotiation that would take a large company to contract for a payroll service.
- The component services and the composite applications built from them are mutually *interoperable*.

In a virtual service oriented grid environment, the service components mentioned above may not be designed to behave as full fledged applications, but intended to be used as building blocks for applications. We will use the term *servicelet* or *microservice* to specifically denote service building blocks.

To bring another example, the outsourcing of software projects usually requires the same painstaking and lengthy process for both the service requester and the service provider.

In the 1990s the Internet unlocked trillions of dollars in value through *disintermediation*, essentially directly connecting producers and consumers and removing the middleman.

Ironically, service orientation has the potential of becoming even more momentous. It swings the pendulum in the opposite direction in a process of *reintermediation*. Only this time the middlemen will not be humans, but machines talking to each other through the service abstraction.

In this environment, it becomes possible to monetize a service that was initially custom built for use at a particular organization. Services become an encapsulation of intellectual property, a unit of value, discoverable and interoperable through standard interfaces.

As the market for services evolves and matures it is not far-fetched to see these services traded as commodities and sophisticated instruments developing, such as futures markets.

In this brave new environment the upfront cost of building a service component in the first place can be amortized over multiple instantiations. This reuse lowers the per instance cost of a service, which in turn lowers the implementation cost of the business application where the service is used. For service consumers, this is how the traditional cost benefit of SOA based applications is realized. By placing a service in the marketplace that perhaps initially was intended for in-house use, service purveyors have the option of generating additional revenue streams to offset development costs. Hence service components are no longer items in the cost side of the ledger. They can become revenue generators.

Compound applications built through servicelets will also become self-optimizing through metrics implicitly or explicitly defined in a service's service level agreement (SLA).

The consideration to SLAs in compound applications in both internal and external interfaces will naturally enable factoring in other nonfunctional requirements, such as regulatory compliance or quality and performance standards.

Consequently, in parallel with the disintermediation and reintermediation there will be a process of *disaggregation* and *reaggregation* of applications and services. As mentioned above, the granularity of the constituent components of an application will become far smaller than what is seen today. For instance, a payroll application is usually a single logical function outsourced to a service provider such as EDS. A virtual service oriented environment will allow an organization to mix and match a combination of business logic software, the data, the storage on which the data is hosted, the database used to access the data, and the servers on which the application runs.

Developing applications in a service oriented ecosystem will require a marketplace where a broad range of servicelets is available providing application architects and engineers with a broad palette to choose from to fit specific goals. For instance, it is possible to select servicelets encapsulating storage and server hosts in distributed locations for the purposes of disaster preparedness.

The servers used to build this application may actually be virtualized instances, not physical servers. The service consumer does not care or may not even be aware of whether the server instances are physical or virtual as long as the provider meets pre-agreed SLAs for performance

and security. Universal interoperability will foster a market for commodities and futures for certain popular service components.

Conversely, servicelets need not be pure software entities; some functions can interact with the physical world: when a student at some university uses a course registration application, the registration process invokes a servicelet to purchase the corresponding textbook through a book supplier such as Amazon.com. The invocation of the Amazon.com Web service interface triggers a chain of events whose end result is a book mailed to the student's address.

The event chain may be extremely complex. For instance, the actual seller of the book may not be Amazon.com but one of the associated resellers. Throughout this chain information flows as needed to make sure the cost settlement is appropriate for each party according to the agreements in place and that the book is eventually mailed to the right address.

## **An Industry Example**

Because of the expense associated with negotiation and teardown of business applications, these applications today tend to have a long life. Teardown may be a complex process involving the removal of on the premises applications and equipment and negotiating a new contract with an alternative provider. A virtualized service environment will make dynamic behaviors more practical. Decision makers will be able to gather supporting data in a fraction of the time considered possible today. An application could more easily be built to address a specific question or support a short duration campaign providing increased business agility.

Let's take the example of a food industry marketing analyst interested in performing a study on purchase patterns of a certain line of products using business intelligence (BI) methods.

The analyst licenses a customer loyalty database from a grocery chain company. The grocery chain is willing to grant such licenses to third parties as part of a strategy to create additional revenue streams from intellectual property generated from business operations.

In looking at the grocery chain, virtualization and service orientation will lower the barriers for mature industries with a vision to rediscover jewels of intellectual property they had all along in the form of data and processes, and find out that they can enhance their revenue in the services marketplace.

Conversely, service consumers may find it more economical to integrate servicelets from these companies rather than spending years in developing and honing the processes and knowledge internally.

The analyst rents a one-use instance of the customer loyalty database and goes to an application service provider (ASP) who orchestrates the following actions:

- Pulls out one encrypted copy of the customer loyalty database
- Purchases storage from a storage services provider in China to house the database
- Purchases a one-week use license from the services organization of a database independent software vendor (ISV). The database ISV turns around and rents two servers from a data center service provider for a week to host this run.
- Purchases a one-week license from the services organization of a BI ISV.
- Sends a single bill to the analyst's department.

The analyst issues queries from her desk and completes the week long research project. The ASP, as orchestrator, destroys all copies of the database in a provable manner and to the satisfaction of the data owner when the project is complete and the transaction is closed.

Here are some observations for this scenario:

- All data travels encrypted and compressed until it gets decrypted in memory. The only way to snoop would be to attach a digital analyzer to the processor bus. Hence service provider hosts are not even aware nor are they allowed to snoop on what their customers run in their data centers.

The servers are tamper-resistant certified by third parties taking advantage of the pre-existing trust chain. An independent security services provider monitors and certifies data centers for a number of customers around the world.

If an attempt is made to open a cabinet during a sensitive run, the server is automatically shut down. Too many of these incidents may lead to the data center owner losing its license to run external jobs. This environment allows deployment of applications from resources available worldwide.

- All communication is encrypted requiring mutual authentication, respecting the requirements of each participant in a compound

application. Interaction between mutually suspicious partners is always assumed.

The philosophy for these security measures won't be much different from communication taking place today in a potentially hostile Internet. The level of privacy and security is important for a reliable bill-back and cost settlement function.

The mechanisms also support tamper resistant system management actions that can travel across different organizations: when the infrastructure for this experiment is decommissioned, the database owner gets a proof that all copies of the database have been destroyed.

- Each player gets a view of a managed resource appropriate to their function: The host owner gets a view of the physical resources. Most requesting customers get a view of virtualized resources. In practice, differences between physical and virtual resources will be blurred.

This is a scenario for a highly dynamic, real-time, pay-as-you-go infrastructure. For companies highly variable seasonal usage it may be cheaper to rent than to host servers that are near idle all the time.

## What's in Store in This Book

The chapters in the book are structured as follows.

Chapter 1 covers basic definitions, and specifically the components of the virtualization, service orientation, and grid computing triumvirate and their synergistic relationship.

Chapter 2 describes the business environment that is fueling the demand for virtual service oriented grids that is at the same time making them economically feasible. Paradoxically, we see increasingly automated technology that presents an increasingly responsive, agile and personalized face to the customer. This evolution is driven by IT managers' quest for value, in terms of cost, scalability, agility, reliability and a slew of other abilities while improving service.

Chapter 3 provides insight into the architecture of virtual service oriented grid systems: how they are built, what they are good for, what are their limitations.

Chapter 4 covers the technology building blocks that make the architecture described in Chapter 3 possible.

Virtual service oriented grids are enabled not only through advanced technology. A necessary element is the upfront and tireless work done by standards committees to ensure service to service interoperability, covered in Chapter 5.

Chapter 6 covers strategic considerations for deploying a virtual service oriented grid infrastructure in its different manifestations as well as integration issues that arise in combining the building blocks described in Chapter 4 to build the technology vision described in Chapter 3.

In Chapter 7, we examine a number of case studies on virtual service oriented grids and understand the dynamics behind their industry adoption—or failure—and trace their evolution in the past, present, and future.

In a shift from an expository discourse in the previous chapters to a prescriptive style, Chapter 8 puts the limelight on the reader, using the foundation laid out in the prior chapters to help the reader build a grid strategy aligned with her or his particular business needs and conditions. Examples and case studies are provided with this chapter.

Finally, in Chapter 9 we button up in a succinct manner the concepts developed throughout the book.

## **Acknowledgements**

Integration, collaboration and synthesis are essential to SOA, and the content presented in this book reflects this spirit. The content reflects the experience and the creativity of many people and industry giants, perhaps under a new light. This book would not have been possible without their contributions through their written work, technical discussions and even casual conversations and business meetings.

The authors have calibrated these insights with the experience from professionals at Intel IT through numerous joint projects spanning many years. The authors acknowledge their openness and willingness to share that experience, the duress of extreme work and budgetary issues typically of the industry notwithstanding. Some of the fundamental ideas in the book came from a strategic collaboration between Intel Solution Services and the Enterprise Architecture efforts led by the Intel Chief Architect and Vice President at the time, Prasad Rampalli. Today Prasad is the Vice President of the Digital Enterprise Group and Director of End-User Platform Integration for Intel Corporation.

The contribution of Stephen S. Pawlowski, Intel Senior Fellow and CTO for Intel Digital Enterprise Group is also acknowledged as the

sponsor and initiator of this book project. The support of Intel Press Publisher Richard Bowles throughout the project has been greatly appreciated.

The support from Rick Echevarria, Vice President for Intel Sales Marketing Group and General Manager for Enterprise Solution Sales is also acknowledged.

Jason Devoys, Qui Hoang, Jim Hobbs, John Morton, John Baudrexl, and Keith Uebele provided a technical and business evaluation of the book content. Keith in particular went through most every chapter in the book. His deep and insightful comments have made the book a much better product.

The authors acknowledge the contributions of the external reviewers, Dr. Jun-Jang Jeng, IBM TJ Watson Research Center, Dr. Wei-Jen Lee, Director of the Energy Systems Research Center at University of Texas, Arlington, Dr. Gerald Sheblé, Maseeh Professor of Electrical and Computer Engineering, Portland State University and Dr. Shimon Y. Nof, Director, PRISM Center, Purdue University. Other reviewers include Jim Richmann, James P. Hobbs and Tarasov Artem.

Special thanks to project manager Bruce Bartlett and editor David Clark who kept the project on track and the authors motivated through the gentlest but persistent nudges. The production team took the raw material and made a great looking book from it. I especially appreciate the work of, Ron Bohart, the artist who created the intriguing cover art.

Kathleen Masterson, Kevin E. Patterson and Shelley L. Rowe performed a pivotal role in getting the word out about the book through their marketing efforts.

For anyone I may have missed, please accept my deepest apologies.

*Enrique Castro-Leon*  
*August 2008*

## **Dedications**

(Castro-Leon) I would like to acknowledge the tireless support of my wife Kitty during the many virtually nonexistent weekends that it took to get the project accomplished. To her, this book is dedicated.

(He, Chang and Peiravi) We dedicate this book to the contributors, reviewers, and Intel Press team who supported us through the development and completion of the project.



