

# The Impact of Connection-Based Architecture on the Future of Business Process Modeling Systems

By Ken V. Krawchuk  
CTO, Business Architects LLC  
September 15, 2006  
US Patent No. 5,418,942, 5,564,119, and 5,960,437

## Introduction

Business Process Modeling (BPM) is simultaneously an innovative, emerging discipline and also a tried-and-true business tool. Its continuing evolution stretches back more than half a century, beginning with primitive flowcharts and reaching to the present day with the XML-based Business Process Execution Language (BPEL) along with the Service-Oriented Architecture (SOA) and other tools that support it. However, despite decades of progress, a brief examination reveals a glaring shortcoming: BPEL is still just another flowchart, replete with the same diamond decision boxes, spaghetti-like connections, and arcane symbology.

The current incarnation of BPM tools can be likened to the products of the automotive industry: certainly cars today are safer, get better mileage, and require less maintenance, but their internal combustion engines are still essentially the same ancient technology, albeit improved, retaining many of their original shortcomings, such as explosive fuel, waste heat generation and dissipation, exhaust pollution, etc. Similarly, today's BPM systems have successfully addressed certain challenges, such as machine independence, faster implementation time, and limited module reusability. However, because of their legacy architectural heritage, BPEL (and programming languages in general) suffer inherent difficulties which continue to stubbornly hold their own, including:

1. The tools are difficult to readily understand, and gaining prowess requires a steep learning curve.
2. The tools are incomplete in and of themselves, and are thus fractured across several discrete products, each with their own syntax and scope, exacerbating the steep learning curve.
3. Combining these factors, the tools are typically beyond the comprehension of the average business user, relegating their utilization to a cadre of expensive specialists. This has generally been the case with virtually all Information Technology solutions. (Spreadsheets and word processors are among the few notable exceptions.)

To move beyond these inherent problems, a revolutionary new direction in system architecture is required, one that does away with steep learning curves, fractured syntax, and specialized practitioners. One such solution is found in connection-based architectures.

## A Brief Overview Of Connection-Based Architecture

A connection-based architecture is a radical departure from the basic premises of traditional programming and modeling techniques. In a nutshell, the fundamental concept is that any information system can be represented *exclusively* by connections, rather than being based on bits, bytes, files, programs, or objects. Several patents<sup>1</sup> related to connection-based architecture already exist, and the

---

<sup>1</sup> "System And Method For Storing And Managing Information", US Patent No. 5,418,942, 5,564,119, and 5,960,437

approach has been employed over the years with great success across a wide variety of applications<sup>2</sup> both inside and outside Information Technology.

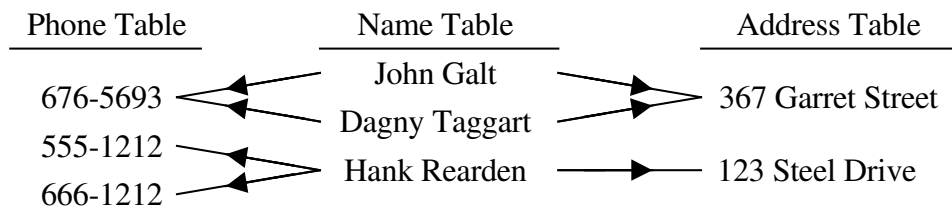
As a simplified example of how a connection-based architecture operates, consider the ubiquitous telephone book. It contains a list of people's names along with their addresses and phone numbers. If a single person has two phone numbers, they have two entries with their name and address duplicated; similarly, if two people have the same phone number, their addresses and phone numbers are duplicated. Specifically:

John Galt, 367 Garret Street ..... 676-5693  
 Hank Rearden, 123 Steel Drive ..... 555-1212  
 Hank Rearden, 123 Steel Drive ..... 666-1212  
 Dagny Taggart, 367 Garret Street .... 676-5693

When this data is represented using older legacy systems, each line from the telephone book would typically be stored as a sequential list using a fixed-length entry in a file, duplications included. Specifically:

Phone Book File		
John Galt	367 Garret Street	676-5693
Dagny Taggart	367 Garret Street	676-5693
Hank Rearden	123 Steel Drive	555-1212
Hank Rearden	123 Steel Drive	666-1212

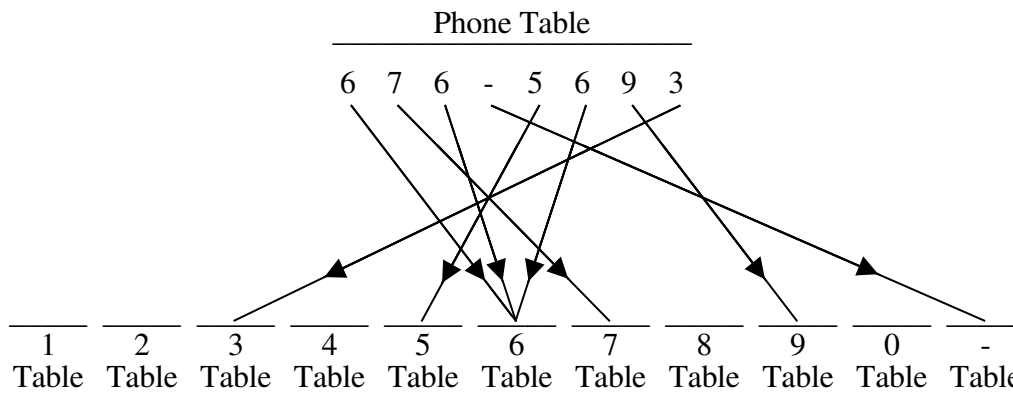
Obviously, this duplication is wasteful of system resources and also a maintenance nightmare. Fortunately, when implemented in a typical database system (or using a connection-based architecture), the duplications can be avoided by normalizing the data by storing the names in one table, the addresses in a second table, and the phone numbers in a third, then using a system-maintained connection to associate each name with its corresponding address and phone number. Specifically:



In virtually any systems, this approach would be considered the final solution and the most efficient use of resources. However, this ignores the fact that other duplicates exist which could also be normalized, specifically, the repeated use of the individual digits and letters which make up the names, addresses, and phone numbers of each entry.

To eliminate these duplications, a series of tables can be built, each corresponding to one letter of the ASCII character set. Each of these tables would contain only that one letter, such as an "A Table" containing only the letter "A", a "B Table" containing only the letter "B", and so on. Then, when adding values to the Name Table, Phone Table, etc., instead of storing the actual character, a connection to that character's table could be used instead. For example, looking only at the first phone number in the Phone Table, the connections would be:

<sup>2</sup> [www.BusinessArchitectsLLC.com/casestudies.html](http://www.BusinessArchitectsLLC.com/casestudies.html)



In this manner, all duplication is completely eliminated because there exists only one copy of each character. Whenever that character is used, a connection is made to its table, rather than using the ASCII character outright. The connection replaces the character.

Taking the concept yet another step further, recall that each character is the only entry in its respective table. For example, the only entry in "6 Table" is the ASCII representation of the digit 6. However, it is important to note that the value "6" does not actually need to be physically present; it is enough to know that a connection is being made to "6 Table" to know that the desired value is "6" irrespective of the table's contents. In other words, the actual ASCII encoding of the number 6 (or any encoding, for that matter) is not required. The ASCII can be eliminated, and the underlying "6 Table" can be left empty. Only the connection to the table is important; again, the connection replaces the character.

Thus, the information contained in the phone book example, or in any database, can be represented exclusively by connections, independent of how those connections may ultimately be implemented. By extension, since any executable program can be encoded as data, processes too can be represented exclusively by connections.

This is the hallmark of a connection-based architecture: all connections, no data.

### **An Implementation of a Connection-Based Architecture**

Connections in and of themselves are not a panacea which solves all the ills of existing BPM systems; the connections reflected in any large or complex BPEL flowchart plainly prove that. What is needed is a disciplined approach which takes advantage of the strengths of a connection-based architecture while avoiding the introduction of potential weaknesses. One such implementation is the Business Architecture System developed by Business Architects<sup>3</sup>. The System capitalizes on the concepts of a connection-based architecture without sullyng its simplicity with unnecessary, complicated frills. The result is a unique, intuitive, easy-to-learn graphical business modeling, automation and management tool featuring a universal language to rapidly create business models that are readily understandable by both technical and business users alike. The System can be used to capture requirements, identify the independent services, design the system, drive its implementation, execute the process, create the system documentation, and manage the overall project effort, all with a single tool using a single syntax.

Because of its simplicity, that syntax can be quickly and easily summarized: its graphical representation consists exclusively of labeled connections, with the label residing within a box, and a

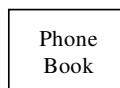
---

<sup>3</sup> [www.BusinessArchitectsLLC.com](http://www.BusinessArchitectsLLC.com)

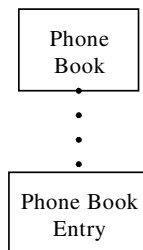
straight line showing the connection. A connection can be either synchronous or asynchronous. For data, synchronous means a one-to-one connection, while asynchronous is one-to-many. For process, synchronous means that procedures are orchestrated in series, while asynchronous procedures are performed in parallel. Synchronous connections are represented by a solid line, while asynchronous connections use a dashed line. The models themselves are hierarchical, resembling an inverted tree where lines extend directly from the center of the bottom of a higher-level box down to the center of the top of one or more subordinate boxes. Where a range of possible choices is involved, such as an enumeration of data objects or a selection of a processing path, the line is not straight, but rather squared off, and is referred to as a case connection. Add an iteration symbol to indicate a repeated process, and this constitutes the entire syntax of the Business Architecture System. What other BPM system can have its syntax described in a single paragraph? This simplicity translates directly into ease of understanding and use.

When using the System to define and capture requirements, a top-down process known as "symbolic decomposition" is performed, where data and process objects are broken down into their component parts. To illustrate how the Business Architecture System can be used to define data, the aforementioned phone book can serve as a fine example.

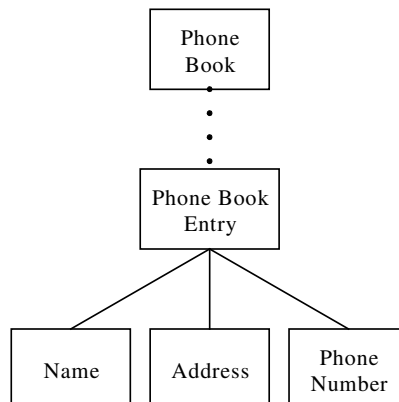
Starting at the top, the phone book itself is represented in the System as a labeled box, as follows:



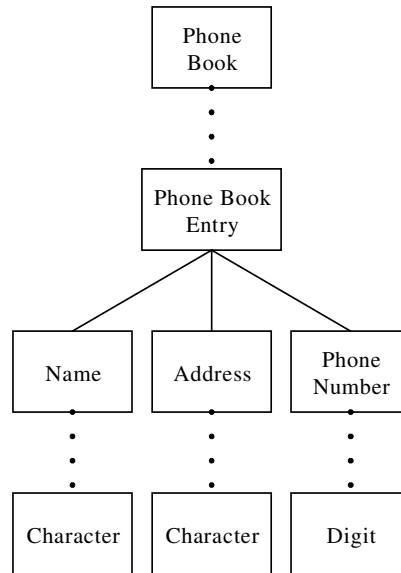
The model is then expanded using symbolic decomposition to divide a phone book into its component parts. Since a Phone Book consists of many individual phone book entries, it would be modeled as follows:



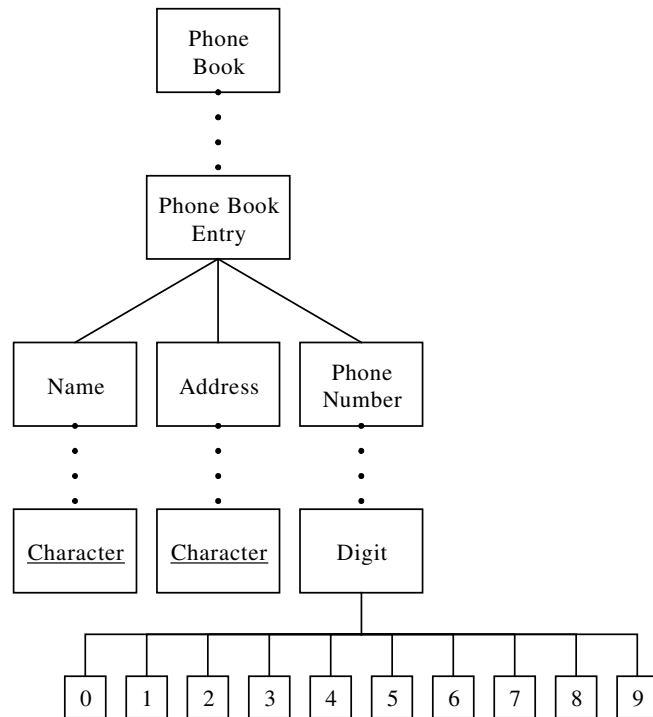
Continuing the decomposition, each Phone Book Entry consists of a single name, one address, and a phone number:



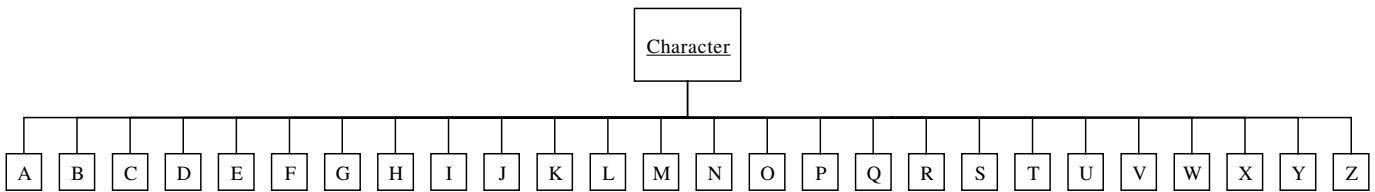
Next, Name and Address each consist of many characters, and Phone Number can have many digits:



Decomposing Digit into its component objects expands the model further by adding case connections to enumerate all possible values for Digit:



A similar decomposition is also performed on Character; however, within the Business Architecture System environment, multiply-used objects such as Character are hyperlinked to a common definition, as shown:



This completes the entire model which defines the data structure of a phone book. Note how its description lacks any mention of the underlying technology upon which it may rest, yet there is sufficient information to describe and implement the phone book. The easily-understood model can be reviewed by the business process owner to verify its completeness and accuracy, then handed over to a legacy programmer to be translated into any language, from FORTRAN to Java, or it can be used directly within its native BPM system, the Business Architecture System. The System assigns values to this definition by connecting occurrences of the data to the respective definition boxes, and, as with all other aspects of a connection-based architecture, these values themselves are composed exclusively of connections.

Process models are built using the same syntax and decomposition techniques, and are just as intuitive as the data model. The process and data models are connected in a disciplined fashion to form a blueprint of one or more business processes that describe, monitor, and manage the definition, implementation, and execution of any subsystem, application, business, or enterprise.

### **Benefits of Connection-Based Architecture**

There are a plethora of benefits that result from employing a connection-based architecture as the heart of a BPM system, programming language, or operating system. Some of these benefits overlap the advances already made with recent BPM systems, while others stretch far beyond what is currently available. These benefits include:

#### *Machine Independence:*

At the most basic level, the universe of primitive operations in a connection-based architecture is extremely limited, encompassing only four actions: Hook, Unhook, Locate, and Compare. Using the Hook primitive, definitions and values can be added; using Unhook, they can be deleted. Combining the two allows for changing a value. The Locate primitive is used to "walk" connections, and Compare allows for choices when walking. Using only these four primitives, entire systems can be constructed. Because there are so few primitives, any user-developed system is eminently portable. Since only four primitive "drivers" are needed for any given hardware platform; and because of their simple nature, accommodating new platforms becomes almost trivial, and the classic problem of platform-specific operational anomalies can be eliminated entirely.

#### *Infinite scalability:*

Since all information is represented exclusively by connections, this yields infinite scalability in all dimensions, assuming only that physical storage space is available. At the detailed end of that scalability spectrum, a value can always take on more digits or characters without any alteration to the models, and the field definitions never run out of architectural space. In the middle of that spectrum, another record can always be added to any set of data, no matter how large the record or the set. At the

high end, its bus-like Service Oriented Architecture can clearly represent and efficiently handle an infinite number of services. No existing BPM product so completely frees the user from concerns over scalability. Their dependency on legacy architectures assures the user that they can one day run out of space somewhere, either by truncating a value, filling up a file, or overloading or exceeding the limits of the architecture. The Business Architecture System eliminates this possibility.

#### *Universal language:*

A major part of the power of the Business Architecture System is rooted in its complete lack of technology-related issues and attendant technical concerns. Its universal language is quick to learn, easy to master, and requires no technical knowledge to implement even the most complex of systems. Rather than introducing an ever-growing host of cryptic symbols, confusing syntax, mind-numbing XML, or other barriers to everyday use, there are only a handful of graphical symbols which can be used to model any system. Further, the data and processes of existing legacy systems can also be modeled, allowing for seamless access to any system or data anywhere without any concern for its underlying technology.

Every aspect of the Business Architecture System relies upon its graphical language, whether the models are related to data, process, project management, document design, or any other aspect of a business model. Not only does this make the use of the System extremely intuitive, it also provides a clear, concise means of communicating business rules to all levels of an organization, from corporate management to the business process owners and their staff, to the auditors, to the technical team.

The typical BPM system is a disparate collection of products which, taken together, perform business process modeling, with each discrete product targeting different process types, life cycle phases, and user types. No one product simultaneously addresses so many needs of the BPM market except for the universal language of the Business Architecture System.

#### *Empowering the business process owner:*

In the early days of personal computing, few could confidently brave the vagaries of the DOS operating system and its cryptic, often-frustrating mode of operation. Introduction of the Windows operating system eliminated this hurdle, giving a great number of people access to computing power for the first time. However, while Windows opened the door to the use of a wide variety of end-user applications, it did little to extend to them the power to create or modify those applications. But when a simple, intuitive syntax is coupled with a technology-free modeling system, the barriers to programming a computer vanish. Witness what non-technical people have already accomplished with the almost-unintelligible syntax of Excel formulas, SQL queries, and VB code; imagine how much more they could accomplish with a programming environment which can be readily understood after only a few minutes instruction and permits productive activity within mere hours. Business process owners and other stakeholders can take personal charge of their systems and data responsibilities. In those cases where they choose to delegate that responsibility, the final, executable models are so readily understandable that their fidelity to the business rules can be easily verified and confidently counted upon.

#### *Unambiguous specifications:*

Too often, the requirements specifications are insufficiently clear to insure a project's success. The underlying reasons for this are legion, and on occasion the root causes may not even be under control of the business. In some cases, the business process documentation may simply not be up to date. In

other cases, the business knowledge may reside only in someone's head, or worse yet, the knowledgeable person may have moved on to a new assignment, taking the business knowledge with them. In extreme cases, the participants may not even understand their own business. But too often the cause can be attributed merely to a lack of proper requirements definition tools.

The deleterious effects of incomplete requirements specifications can manifest themselves in a number of undesirable ways. If the problem is caught early enough, the impact may result merely in a project delay as the team waits for the requirements to be clarified. In cases where the schedule is tight, often the only way to remain on target is to scale back on the expectations and functionality of the system. The lack of complete specifications can result in a system that does not meet the needs of the business, and in the worst-case scenario, the business may not even be aware that the specifications are incomplete until too late.

Using the Business Architecture System, requirements are unambiguously captured early in the effort. In those areas where a project's requirements are incomplete, any weaknesses in the project definition are highlighted by the System, driving the discovery questions that ultimately complete the business model. Using an iterative approach, business rules are captured in a dynamically-navigable, graphical model that is easily understood by management, business process owners, and technical staff alike, serving as a business knowledge repository that survives any individual.

#### *No more bulky tomes as specifications:*

Traditionally, requirements specifications are captured or rendered as text; and the more complex the system, the bulkier the results. These tomes are not only difficult to write, but also time-consuming to review, especially in repeated rounds of updating; and the larger the document, the higher the likelihood that it would not be adequately reviewed for accuracy. Worse yet, the text is often open to interpretation, and in some cases impossible to decipher, compounding the difficulty of obtaining a confident signoff.

Typically these documents are incomplete, subject to misinterpretation, or just plain wrong, leading to all the problems usually associated with any incomplete specifications, such as project delays, systems that do not meet the needs of the business, or outright project cancellation. Often they are simply set aside once the project progresses into the design phase, bringing into question the fidelity of the business rules to the implemented system, especially in situations where English is not the native language of the composer, reviewer, or implementer. The end result is a long, tedious learning curve for anyone new to the system.

Using the Business Architecture System, these textual behemoths can be eliminated. Instead of using text that is time-consuming to compose and read, as well as open to misinterpretation, the System captures all of the business requirements as graphical process and data models that are concise, simple to understand, and easy to access, in a dynamically-navigable format. The models form an unambiguous definition of the business process that serves as a tool for complete understanding of the requirements and a better understanding of the overall business.

#### *The end of dead-end specifications:*

In many cases, the specifications or business rules reside in a separate document or software product that is not tightly integrated with either the definition, design, or implementation of the business process. This leaves the specifications document an end in itself, and once completed, the process of keeping it up to date is a tedious one that is seldom performed. In a traditional environment, the

specifications become a historical, dead-end document that is only glancingly referred to during subsequent phases of the project. Some BPM systems integrate the generation of specifications with the subsequent design and/or implementation, but the format and syntax of the business rules change from one phase to the next, usually radically, becoming increasingly complex and increasingly technical until the end result bears absolutely no resemblance to the original.

Whenever the design or implementation is not integrated with the physical specifications document, there is always a very real doubt as to the fidelity of the final system to the original business intent. Even when there is some degree of integration, the shifting syntax introduces similar doubts, and, more importantly, detracts from understandability.

With the Business Architecture System, the specifications are not tossed aside when completed, but rather expanded upon to form the system design, which in turn is further expanded to create the operational system. The common thread of the System's universal language insures that the same business intent that is captured unambiguously during the specifications is identical to what is ultimately implemented. This singularity insures that the implemented process is just as understandable to all stakeholders as the original specifications, from C-level management to the business process owner to the implementers.

#### *Mitigating ineffective or inexperienced project leadership:*

The best specifications are of little value if the leadership is not capable of managing the process of transforming policy into practice. But frequently, many managers lack sufficient experience to oversee large, complex projects, or sometimes even the simple ones. Some are unfamiliar with any sort of efficient project management techniques, while others fail to understand the tradeoffs involved in resource allocation.

False starts, spinning wheels, and poor resource management lead to cost overruns and schedule delays, while lack of management experience can result in unrealistic expectations, insufficient risk management, and project discord. Poor communication or a lack of status reporting skills can mask a troubled project, leading to rude awakenings. In the end, senior management must settle for a late, expensive, less-than-optimal project, or replace the project manager and risk a disruption in project continuity. In all of these cases, the project suffers.

Using the Business Architecture System, any project manager can better understand how a project is structured, resulting in more-efficient resource allocation and more-accurate status reporting. The “factoring” ability of the System provides management with a detailed roadmap for resource allocation, and empowers the individual system designers by providing them with a clear scope of responsibility for their portions of the project. The System also highlights weak or incomplete segments of the project design and specifications, providing any manager with all of the tools needed to successfully complete their project. This helps develop a manager’s skill by providing a tool to keep abreast of the current status of specifications, design, implementation, and testing.

#### *Effective management of dispersed or global teams:*

Managing dispersed teams and offshore resources add to the difficulty of any undertaking. Language barriers, time differentials, and lack of face-to-face communications all conspire to warp any project plan. Integrating any widely-dispersed team is a challenging management exercise in itself, aside from any difficulties in the actual project.

As with poor specifications, the mismanagement of dispersed teams results in late projects, scaled-back expectations, or even the failure of the project or enterprise. The possibility of unexpected problems rises dramatically when management is not intimately involved with the day-to-day activities of the team members, or conversely, when team members run off in a different direction than called for in the project plan. Distance only makes these challenges more pronounced.

Fortunately, the Business Architecture System structure is inherently optimized for managing dispersed, diverse teams. The blueprints generated by the System delineate clear areas of responsibility and, more importantly, the interfaces between those areas, allowing for a more-compartmentalized distribution of responsibility. The universal language used by the System spans cultures, virtually eliminating the possibility of misinterpretation inherent in textual specifications, and allowing senior management a direct portal to ascertain the progress, status, or shortcomings of the project regardless of language barriers or distance.

#### *Stopping scope creep:*

One of the most common reasons why projects fail is poor management of scope. New requirements are routinely piled onto projects at all stages, and each one adds its own complexities, resource requirements, and retrofitting to accommodate them. Often these changes are not adequately reflected in the project plan, if at all, due to schedule pressures, management concerns, or other reasons external to the project.

Scope creep makes any schedule or project a moving target. As goals shift and morph, expectations quickly become confused, blurring the definition of project success. Budgets, schedules, and resource allocations are strained to meet the unforeseen requirements. The responsibility for accommodating these requirements is seldom clear cut, leading to potential discord between the project, its staff, vendors, and senior management. Unless checked early on, the inability to manage scope can easily doom any project, or give even successful projects an aura of failure.

Using the Business Architecture System, the original scope is captured very early in the project in unambiguous terms, helping to define expectations and defy inadvertent changes; and when requirements do inevitably change, there is little or no question of where responsibility for those changes lies. Should a project be threatened with changes of scope, the models created with the System allow for a quick assessment of the impact of the change on the project design, schedule, and resource allocations, as well as fostering a project-efficient solution. Changes of scope can then be rationally assessed as to their efficacy, desirability, and impact on the project rather than on speculative, subjective guesses.

#### *Avoiding a perceived lack of contributions to business goals:*

There is a misconception that some CIO's tend to embrace technology solutions over business solutions, subsequently bringing into question the overall ability of technology to adequately meet business goals. Many of these technological solutions are more often organized around Information Technology disciplines rather than services of value to the business, and couched in terms not readily understandable to upper management or business process owners, if at all. The situation is ripe for the creation of an IT stovepipe, or at least the appearance of one.

The most disastrous result of these practices is a loss of credibility for Information Technology, and the very real perception that IT does not actively reflect management goals. Budgets are questioned, improvements are warily embraced, and justifications are not always understandable.

One of the most significant benefits of the Business Architecture System is rooted in its focus on business solutions. Its universal language effectively represents the underlying technology in understandable terms, allowing all levels of management to concentrate on providing business solutions rather than technological solutions, making the CEO and CIO active participants in developing and meeting business goals, not a spectator.

#### *Managing to a plan:*

It is a cliché in the Information Technology industry that there is never sufficient time to complete a project. Milestones and objectives in a project plan routinely fall by the wayside as projects progress, adding to the difficulty in managing the project.

The impact of missed deadlines ripples through an organization, fostering an ad hoc project management philosophy as unanticipated resources are allocated and brought to bear. Too often, the project cannot readily absorb the additional resources, leading to their inefficient allocation. Worse yet, the project plan may not be able to provide any guidance for where the added resources may be best used.

Using the Business Architecture System, the project plan is inherently integrated with the system design. The factoring ability of the System automatically breaks the effort into discreet, independently-implementable segments, allowing for greater parallel effort and the ability to concentrate resources on the most critical portions. No other system development methodology handles a massive influx of resources as gracefully.

#### *Business is no longer curtailed by technology requirements:*

Sometimes the needs of technology end up driving the business, rather than the needs of business driving the technology. There are many reasons for this, and they can be summarized in two corollary truisms, that business people often do not understand technology, and conversely, that techies often do not understand business issues. While there are occasional exceptions to this rule, business people who understand technology are expensive, and techies who understand business needs are rare.

The result is a disjoint between business goals and their attainment, often with little advance warning that the goals are not being met. Even when there is warning, the technological verdict is often, "it can't be done" when IT focuses more on the details of the process rather than the desired results.

Another prominent feature of the Business Architecture System is its ability to describe any project or undertaking in non-technological terms. Specifications produced using the System are as readily understandable by top level management as by the programmers and everyone in between. Its universal language allows for business solutions to be documented in complete detail as codeable specifications without any regard for the underlying technology. The System acts as a Rosetta Stone to allow business and technical people to communicate and understand one another.

### *Managing multiple technologies and introducing new ones:*

The Information Technology industry can be counted upon to introduce a constant flood of new software products and methodologies. While each new offering has its own compelling reasons to adopt it, managing an amalgamation of state-of-the-art technologies intermixed with legacy systems is daunting at best.

Seamless integration of systems often becomes an insurmountable problem, resulting in even more technology being brought in to address the interfaces between the systems. With each additional system brought in, the difficulty of managing them increases.

The technological independence of the Business Architecture System makes managing multiple technologies transparent. Since the business rules live outside of technology in an easy-to-understand, universal language, the underlying technology can be readily adapted to fit the business requirements rather than the other way around. Legacy technologies can be "unplugged" from the System and the latest wonders integrated in their place, all without impacting the definition of the business problem being solved, allowing for faster, and more efficient adoption of new technologies as they emerge.

### *Taming temperamental technology:*

Once a system has finally successfully overcome the specifications, design, and management challenges, there are still additional problems which may rear their ugly head when operating the finished system. Programs often crash in the middle of the night or other inopportune moments, with the root cause being bad data, leading to illogical inconsistencies such as multiplying two letters together or encountering other garbage data.

The impact of service interruptions on the business can range from a minor nuisance up through a major catastrophe. The unexpected nature of these glitches impacts not only business goals, but also employee efficiency when they are diverted from their current responsibilities to address an errant program. Middle-of-the-night phone calls only exacerbate the situation.

The Business Architecture System avoids these pitfalls by means of its encapsulation abilities. Bulletproof data type checking eliminates the possibility of bad data entering the system, and once inside, that same feature prevents its accidental corruption. Strong data typing also simplifies the design and implementation processes by prompting the designer to ensure the data is of the right format and level before it comes into the System. This translates directly into more up time and less maintenance.

### *Technology-free access to any data or process:*

Another unique aspect of the Business Architecture System is its complete lack of technology-related issues and attendant technical concerns. Its universal language is quick to learn, easy to master, and requires no technical knowledge to implement even the most complex of systems. Rather than introducing an ever-growing host of cryptic symbols, confusing syntax, or other barriers to everyday use, there are only a handful of graphical symbols which can be used to model any system. Further, the data and processes which exist in legacy systems can also be modeled in the Business Architecture System, allowing for seamless access to any system anywhere without any concern for its underlying technology.

While many BPM systems claim technology-free access to data, none can deliver upon that promise with the ease and seamlessness of the Business Architecture System

## **The Future of Business Process Modeling Systems**

Connection-based architecture has raised the bar dramatically for the expected functionality of a BPM system, and as a result, the Business Architecture System is poised to become formidable players in the evolving, fast-growing BPM market space. The System's connection-based architecture has already proven itself successful in automation solutions for a wide variety of applications, including financial services, system software, process control, project management, and document design, among others. Existing BPM systems and tools must begin to match the power and capabilities of the Business Architecture System in order to successfully compete in the BPM marketplace. The business process owner will demand no less.

### **About the Author**

Ken V. Krawchuk is the Chief Technology Officer for Business Architects, LLC, a software development company specializing in creating Business Process Management software and solutions. He has over 35 years of broad technical and business expertise in the Information Technology industry. He was the lead inventor on the team that conceived the underlying patented technology used in the Business Architecture System. Additionally, Ken is a freelance writer for the Philadelphia Inquirer and other publications, a professional public speaker, and a former candidate for Governor of Pennsylvania.

Mr. Krawchuk can be contacted at [KenKrawchuk@BusinessArchitectsLLC.com](mailto:KenKrawchuk@BusinessArchitectsLLC.com), 267-386-8200 extension 33, or c/o Business Architects LLC, P.O. Box 1, Swarthmore Pennsylvania, 19081.