Human-Centered Enterprise Architectures

Miroslav Benda, Ph.D.
Enterprise Architect
Boeing Commercial Airplane Group
P.O.Box 3707, M/S 6H-WW
Seattle, Wa. 98124

Introduction

This paper deals with the problem of structuring the relationship between computing architecture and the architecture of business processes. Structure follows strategy. The strategy that I take is inspired by the notion of pull systems for manufacturing production; I ask these questions:

1. Can we define a pull system between the information system of an enterprise and its business processes?

2. If so, will we achieve improvements as dramatic as pull systems are achieving in manufacturing?

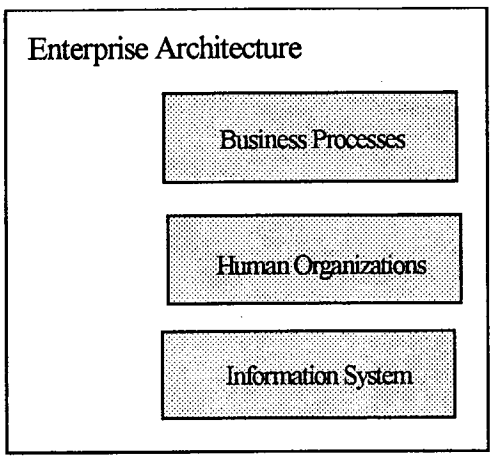
The answer to the first question generates new paradigms for the interaction of business processes and the information system.

With respect to the second question, I have no doubt about the significant improvements. Having been involved in "pushing" technology, the analysis contained in this paper helped to make it crystal clear for me why the technology transfer process is so frustrating. I am certain of substantial savings if we align research and technology developments exclusively with business processes and pull technology into them as part of an overall strategy linking information systems and business processes.

Enterprise Architecture

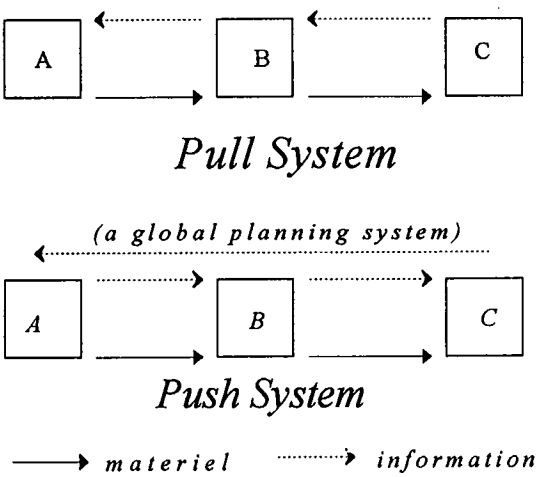
Enterprise architecture is a framework of relationships between business processes of the enterprise, human organizations, and the information system of the enterprise (see [1]). Its purpose is to structure the relationships between people, processes and technology--the three main "components of an enterprise"-- so that the enterprise can effectively react and take advantage of external and internal changes in business and technology environments. Business process and human organizations are enterprise-internal components, while technology is an external variable.

The top level diagram of the three components is:



Pull and Push Systems

For me, the essence of pull and push systems is depicted in the following diagram:



The materiel arrows define the direction of the flow: thus A is upstream from B, and C is downstream of B. In a pull system an activity is triggered by a downstream event; in a push system an activity is triggered by an upstream event and global information. The essential difference between the two systems can be summarized by three properties of pull systems:

Pull system properties:

Locality: Activity in a cell depends only on information received from its downstream neighbors.

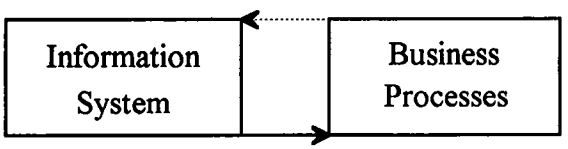
Composition: If A <=> B and C <=> D are pull systems and we define the relationship between B and C so that B is "triggered" by C then A <=> B <=> C <=> D is a pull system.

Insertion: To define a pull system between A and C it is enough to define B and pull systems A <=> B and B <=> C.

Let me now rephrase the primary question of this paper by a more specific question:

To what extent can we make the enterprise architecture a pull system?

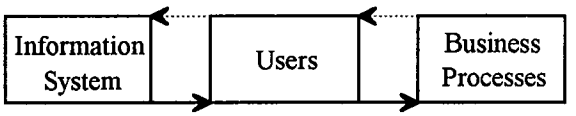
To answer this question we need to place the three basic components of an enterprise in sequence of "materiel" flows. Theoretically, there are six (3!) possible series of these three components. However, pull systems strategy dictates that customers of the enterprise need to provide the requisite pull on the enterprise; the pull is exerted via the products of the enterprise. Hence, the Business Process component has to be placed to most downstream. The information system component then is upstream of this component:



Now, there are just two possibilities to place the third component, humans, into the picture. We place this component between the other two for the following reasons:

The gap between the information system and business processes is enormous and it is difficult to describe the linkage and the flows between the systems. Moreover, unlike in manufacturing, the information and the process systems are not bolted to the floor like machine tools, they are constantly changing - what is even more frustrating is that they are changing at different rates, there is a shearing force between them, but this is a different story.

This is where the principle of insertion comes into play. What we need is at least one fairly constant intermediate stepping stone for which the flows can effectively act as a pull system. While the information system (and processes) have changed drastically in the recent past, human mental and physical capabilities have been constant for a number of millennia and probably will be constant for several centuries to come. Human capabilities are an absolute constraint on both the information system and business processes: no one can rivet with his right hand while programming with his left; humans do provide the required "bolting to the floor". The sequence of components, therefore, is:



The question now becomes whether we can define "material" and information flows so that the resulting system becomes a pull system. This is an iterative process; a reasonable cut at the first iteration is to go first downstream following the "material" flows, from Information System to Users to Processes, and then upstream, Processes to Users to Information System, defining the information triggers. As a navigation aid for the journey I shall rely on an analogy with manufacturing where the notion of pull systems originated. The flow of materiel in this situation starts with the information system and we need to go to the next level of detail here.

An information system consists of three components: applications, data, and the delivery system. These components correspond to components in manufacturing systems.

The delivery system provides the information processing power; it is a utility very much like the electrical power or water supply utility in manufacturing. It is interesting to compare the trend from centralized processing to distributed computing with the transition (in the early 1900s factories) from a central source of mechanical power distributed to individual machine tools by belts and crankshafts to machine tools with their own electrical motors.

Data (and information and knowledge) are what an information system accepts as inputs and what it produces. The terabytes of data containing the product and process

definition of the Boeing 777 for example, are a product of the 777 information system.

Data is "worked on" by applications. Applications are used by the users of the information system to acquire, generate, process, and store data and information. They clearly correspond to machine tools that process "material". To keep the correspondence of applications and machine tools visible I often refer to applications as information appliances.

The table summarizes the analogy.

Manufacturing	Information System
electricity, facilities	*the delivery system*
materials, components	*data, information*
lathes, drill	*applications*

Simplifying: To cover all aspects of the situation we should do now is to divide the discussion into three separate flows: one corresponding to the delivery system, one to data, and one to information appliances. The approach to analyzing the application of pull systems on enterprise architecture can be illustrated by any one of these flows. I choose applications, or information appliances, because that flow has the right degree of complexity. A few words about the other two components, the delivery system and data:

With respect to delivery systems, pull mechanisms are easy to imagine because we are already dealing with physical devices. Delivery systems provisioning actually works like the quintessential pull system, the supermarket: the taste for potato chips pulls the production in the fields of Idaho very much like the demand for computer chips pulls the production of silicon from Oregon sand dunes.

The situation with data is considerably more complex. Even a cursory analysis shows that there is a need for another level of distinction to be able to deal with user pull on data in a coherent manner. Moreover, the analogy with manufacturing suggests that it will require a hybrid push/pull system; a global production control system cannot be based only on pull. The extensive literature on hybrid push/pull systems will undoubtedly suggest useful approaches for dealing with data in information systems.

In the rest of our discussion here we concentrate on applications, information appliances.

Information System to Users Flow: Components

To execute their jobs for business processes from the information perspective, users need applications, information appliances. There are two ways of providing them. One is to provide the appliances themselves and the other is to provide components from which the appliance can be assembled. The ease of integration is the determining factor for preferring one mode of operation over another. If the integration is easy then "assemble-to-order" strategy would be most effective; the Microsofts and Lotuses of the world would provide their word processing and spreadsheet engines and we would integrate these into packages useful to our purpose. On the other side of the spectrum, we get appliances custom made for the job, legacy systems, where the integration and the components themselves are custom made and not reusable.

Users to Processes flow: Tasks

First note that the flow is from users to processes and not from users to other users. To coin a phrase, the process is the customer.. And it makes sense; process owners and organizations come and go, but well-defined processes remain.

The value-added provided by the users of the information system to processes has many forms: a CAD drawing of a component, aerodynamics analysis, heat treatment specification for a part, order for spare parts, cost estimate, etc. What exactly is expected from an individual depends on the clarity with which the expectations of a process are defined.

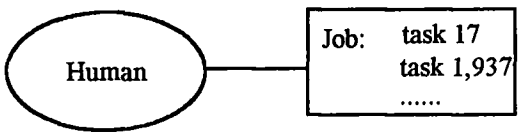
Let us define a task as an "atomic process". A task could be "write a memo", "put a screw there", "compute surface normals"; it is an activity that we will not decompose any further. For a definition of a pull system we need a comprehensive list of tasks. This is the equivalent of materiel and components in a manufacturing setting. My estimate is that at a useful level of granularity we would have tens of thousands of these "components" at the most.

Processes to Users flow: Jobs

As long as there are no changes to processes to which an individual contributes, there is no need to for any visible changes in the application suite at his/her disposal (except perhaps for continuous improvement activity). Changes due to advance in technology should be transparent.

If there is a change in a business processes to which an individual contributes, this may require new information appliances on his/her desktop. What is the information, flowing from processes to users, that would signal the need for a new appliance?

A process consumes resources to achieve its results. An individual contributes to the process by executing selected tasks of the process. Executable processes, in addition to sequencing their tasks, also assign resources required by the tasks; in particular, individuals who are to perform the tasks. Ann individual is typically engaged in several processes. When we collect all the tasks required by all processes to which an individual contributes, we obtain a package of tasks that I call a job. the diagram below pictures the situation:

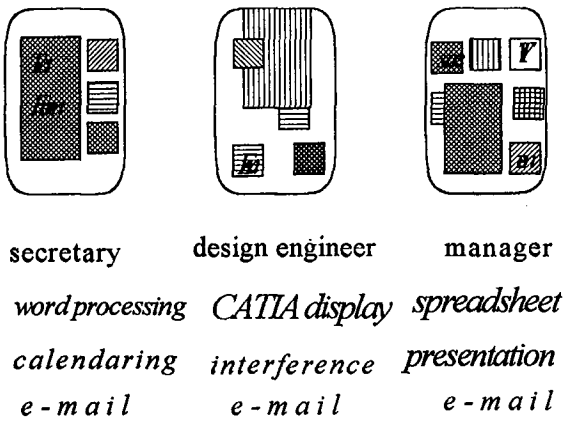


I believe that the number of jobs, i.e. number of distinct task packages of individuals will be an order of magnitude less than the number of tasks. That the number of jobs will be in the thousands.

Users to Information System flow: Applications

In a pull system the information processing tools required by a user to do his/her information processing are determined by the tasks performed that, in turn, are defined by the processes to which the individual contributes. In other words, the list of tasks an individual needs to perform is a shopping list for information appliances. He/she employs this list when "shopping for code"; if he/she does not find the right appliance, or components from which to assemble one, in the "local supermarket" then the user issues a "kanban" to the information system to manufacture the appliance or component. This is where the concept of libraries of reusable components (i.e. information appliance supermarkets), reusable components production lines, brokering, and other market mechanisms come in play).

The signal from a user to the information system for a new application could come in the form of an icon on a desktop screen. The screen would then correspond to the information processing tasks a user needs to perform for the processes he/she contributes to, the screen would represent the job the individual performs (at this moment) for the enterprise. The idea is that those with similar jobs have similar, if not the same, desktop screens. The individual icons would not represent specific packages but information processing capabilities required by the user. For example, the secretary's desktop contains an icon labeled Word Processing; this icon does not need to change no matter what happens to Microsoft Word or Lotus AmiPro. This icon is, in essence, a kanban to the information system to provide word processing capability to this desktop. It is then a matter of the information system, the plumbing and the wiring, to provide an infrastructure that can efficiently satisfy this pull.



A desktop of a user defines the scope of his/her personal digital assistant (an electronic associate). A personal digital assistant is an integrated collection of information appliances that provide the user with the needed information processing capability. (An aside: a personal digital assistant corresponds to a toolbox of a single mechanic. We are introducing community toolboxes on the assembly floor; would a workgroup digital assistant be a better concept than a personal assistant?). The applications that a personal digital assistant contains correspond to the tasks to be performed within a job. This implies that the number of personal digital assistant is at most the number of jobs (several tasks may share an application).

Summary

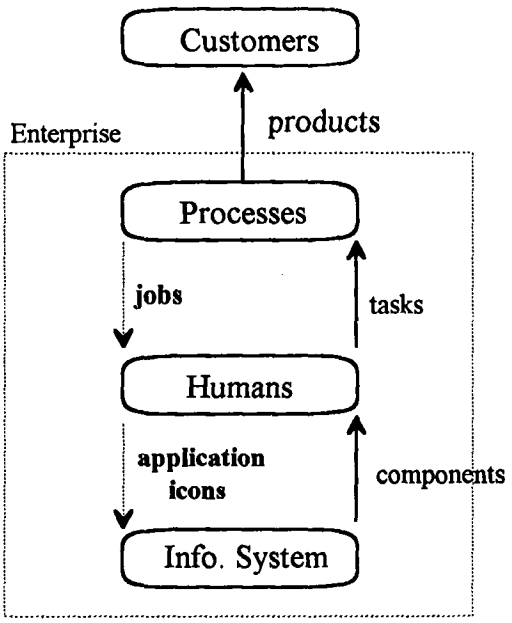
There are different ways of defining components, tasks, jobs and screens corresponding to the jobs. The definition of a task is, or should be, independent of the enterprise; industry neutral taxonomies of tasks that humans perform for enterprises are being constructed.

The definition of components is a field of analysis called Domain Engineering in the reuse of software literature (see e.g. [2]). This is an external variable as well, but an enterprise can exert its influence over the nature of components by building them internally or, given enough clout, by dominating the external market for the components.

The other two variables, jobs and applications, are enterprise dependent. An enterprise can achieve competitive advantage by an appropriate design of jobs and information applicances (applications). This, of course, depends on the

the other two element. Applications are built from components and jobs are collections of tasks.

The diagram below summarizes the relationships; the enterprise dependent variables are in bold:



If we would formulate this as an optimization problem, what would be the utility function? The funtion, clearly, has a number of dimensions. In 1989 in the introduction to the Ninth Distributed Artificial Intelligence Workshop (see [3] p. vi) I defined six enterprise architecture problems. These problems provide some guidance on the features of an enterprise architecture that one would like to optimize; all problems below are stated in the context of an enterprise architecture and infrastructure.

1. Define the degree of centralization and decentralization and determine the optimal balance.

2. Define market mechanisms for internal coordination of information processing functions. (There are some natural candidates for this in the framework above: specifically, components and jobs)

3. Do personal digital assistants provide for better flexibility of the enterprise architecture with respect to technology changes? (it is a fact of life that screens in real enterprises are more constant than even definitions of data elements)

4. What is the design space of enterprise architecture alternatives? (The above framework at this level of abstraction defines seven dimensions for design choices).

5. What are the key engineering concepts to be used in designing enterprise architectures? (E.g., in the above framework, how would we measure the metabolism rate of information, i.e. the speed with which information propagates through the system).

6. What type of architecture is more resilient to viruses and other electronic threats to the health of the enterprise? What are the trade-offs with respect to centralized/decentralized architectures?

If not complete solutions, we need at least some design guidance to these problems. Since the formulation of a problem is half its solution, I believe that by defining the enterprise architecture framework above we have made a progress on these challenges.

References

[1] C.J. Petrie, ed., Enterprise Integration Modeling, MIT Press, 1992.

[2] Domain Engineering Handbook, Software Productivity Consortium report, Herndon, Virginia, 1992.

[3] M. Benda, ed., 9-th Distributed Artificial Intelligence Workshop, Rosario, 1989.