

Value Production in Distributed Real-time Enterprise

Jay S. Bayne

Senior Member ISA, IEEE, ACM
Echelon 4 LLC
jbayne@echelon4.com

Abstract

What is the nature and performance of value production in *distributed real-time enterprise*? Such enterprises are virtual organizations whose missions include the expansion of product and capital market valuations through time-dependent production across a grid-connected federation of cooperating entities. Distributed production requires policies and mechanisms for coordinated command and control in order to effectively operate in such real-time grid-connected federations. This broad research and development area requires at its foundation a dynamic model of distributed value production, a model whose fidelity is sufficient to allow its use across multiple market segments, and that scales from low-level (e.g., manufacturing) processes to high-level (e.g., financial) strategies. We introduce such a model in this paper by developing the concept of *value production units* (VPU).

Keywords: Real-time Enterprise, Global Information Grid, Command and Control, Value Production

1 Modeling Value Production

Enterprises, whether public or private, for-profit or not-for-profit, military or commercial, exist to promote and sustain their *value propositions* in evolving financial (equity) and product market conditions. Value production is therefore a continuous, and typically time critical, computation of those value propositions. In federations¹ of real-time² enterprise³, as well as within the boundaries of its members, value is often difficult to define and measure. Its definition is difficult within enterprises since value propositions, and associated value creation processes, is spread across organizational boundaries and typically involves poorly defined processes. Its definition is made difficult across federated enterprise since joint (mutual) value propositions must be expressed in terms that transcend the participants' own unique and often proprietary definitions of asset and supply chain objectives. Hence, any effort at defining policies and mechanisms for the management and control of virtual organizations requires development of a common lexicon [...], and an agreed upon set of interface specifications to a core set of joint value creation services. This paper presents a framework for discussing value production in federated grid-connected systems that strive to optimize individual objectives while at the same time cooperating on shared objectives.

The model used for this discussion is one based on the notion of "rational agents," semi-autonomous computations that are self-serving (goal seeking) but that participate in alliances as a means to remain viable, to sustain their existence. In abstract terms, a viable system (VS) is one whose value production units (VPU) defines and, operating through well-defined interfaces, executes value production processes. As such VPUs are distributed virtual machines. For our purposes, we shall consider VPU's associated with eBusiness activities, but we could equally well define eScience, eMedicine or other grid-enabled federations.

¹ A *federated* enterprise is one whose participants are semi-autonomous and self-regulating; their designs, following Jeffersonian principles, are required 1) to be viable and identifiable members of a community, 2) to be governed by its laws, and 3) to provide their individual contributions to coherent ensemble behaviors that characterize the outcomes of the mission of the enterprise as a whole.

² Systems are real-time to the extent that timeliness is an explicit aspect of their correct behavior.

³ *Enterprise* is defined as an arbitrary interactive unit of production for systematically creating measurable value through delivery of products or services.

The essential character of eBusiness value creation is captured in traditional accounting terms as defined in income statements and balance sheets. In simplified terms, a VPU is a computation that simultaneously satisfies the need to produce positive returns on invested [capital] assets in an asset chain while profitably providing one or more goods or services to a supply chain. This dual set of objectives is represented in figure 1. As such, value production, when viewed as a black box, is a process with two constituents – *investors* (i.e., suppliers of assets) and *customers* (i.e., consumers of products or services). Along its vertical axis a VPU participates in an asset chain that serves its investors. Along its horizontal axis a VPU participates in a supply chain that serves its customers. A given enterprise may include one or more VPUs. A federation comprises two or more enterprises, each with one or more VPUs.

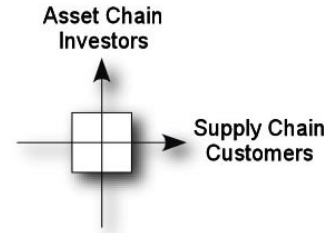


Figure 1 – A VPU

The two-axis model of a VPU supports its asset and supply chains through eight communications ports, as detailed in figure 2. Investors provide [typically capital] assets at port a_i (assets-in) that subsequently yield investment returns on port r_o (returns-out). Customers provide demands for goods or services on port d_i (demand-in) that are fulfilled on port s_o (supply-out).

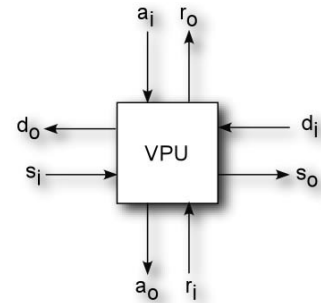


Figure 2 – VPU Ports

A VPU supports two subsidiary channels, one for subordinate VPUs (e.g., intellectual property generation) and one for supplier VPUs (e.g., material stocks.) Subordinate VPUs are allocated investment assets on port a_o (assets-out) that generate returns on port r_i (returns-in). Supplier VPUs receive their demands on port d_o (demand-out) and return their production on port s_i (supply-in).

The horizontal flows are expressed in terms of cost per unit ordered and delivered. The vertical flows are expressed in terms of cost per asset deployed or returned. Costs are measured in (typically, net present value) dollars.

This structure allows VPUs to participate in the production web of a federated enterprise, the two principle threads (chains) of which are shown in figure 3. Each VPU is uniquely identified (i.e., *named*) by its indexed location vertically and horizontally in the grid. Thus, $VPU[k, 1]$ is subservient in the investment chain to $VPU[k, 1+1]$, and is a supplier to $VPU[k+1, 1]$ in the supply chain.

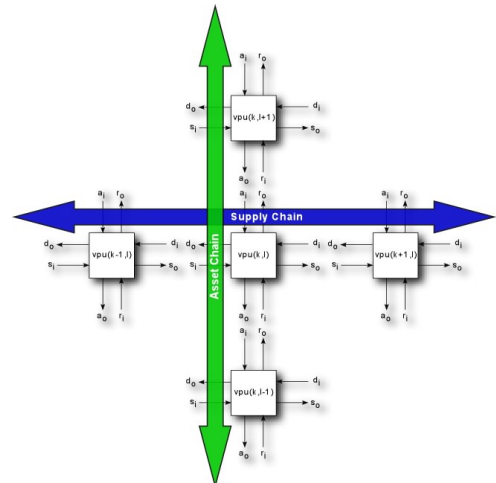


Figure 3 - VPU Production Web

2 VPU Model Behavior

The behavior of a VPU as proposed here will be discussed in some detail, and depends on the notion of performance. Performance, in turn, requires a set of application neutral metrics so that we may compare behaviors within and across enterprise boundaries. So we begin our discussion of the VPU behavior model with a definition of six core measures of performance.

2.1 VPU Performance Metrics

Regardless of its mission, a given VPU's performance may be measured in terms of its *potential*, *capability* and *actuality*. *Potential* is a function of a VPU's design limits. *Capability* is determined by the level of assets currently deployed in realizing this potential. And *actuality* is the current level of real value production given the current capability.

Three additional performance measures can be derived from the first three. *Latency* is the ratio of actuality to capability, indicating how much latent potential remains without additional investment. *Productivity* is

the ratio of capability to potential, indicating how much capability can be added through additional investment. And, absolute *performance* is the ratio of actuality to potential, the present production level given the VPU's design limits.

Figure 4 demonstrates these relations. Initially the VPU is 70% capable with respect to its design potential. At times t_1 and t_2 management action succeeds in raising actual performance (actuality) from 20% to 30%, then 30% to 50%. At t_3 assets are allocated to raise capability to 80% improving throughput. During these periods of adjustment, latent potential (unused capacity) remains moderate at 30% and 20%, while productivity increased from a low of 29% to a high of 81%, a 184% improvement in productivity. At the same time, the VPU enjoys an absolute performance gain from 20% to 65%, an improvement of 220%.

2.2 VPU Timing Considerations

Any discussion of value production units in realizing the objectives of real-time enterprise requires a clear notion of time and consequent timing dependencies with the federated ensemble. The macroscopic behavior of a VPU includes its transport delays, the time it takes for an invested asset to produce a return and the time it takes the VPU to translate a customer demand into a fulfillment. These two transport delays are key elements in considering the temporal performance of an enterprise.

Transport delays are typically expressed in terms of probability distribution functions with certain statistical properties. In figures 5a and 5b the vertical (asset) and horizontal (supply) chain delays are the processing and communications delays of the value computation going on inside the respective VPUs. T_{t+1} and T_t are the processing times for sequentially absorbing assets and subsequently generating a cumulative return to an investor. Likewise, T_k and T_{k+1} are the computational delays in a cascaded production line that accepts an order and produces the requisite scope of supply to a customer.

In a real-time system, these delays are the subject of specification, typically in the form of time constraints. For example, the end-to-end time ($T_k + T_{k+1}$) might have a deadline that is understood by the involved VPUs. As a consequence, each VPU $[k, l]$ represents a computation that must solve simultaneously the timing objectives of the horizontal supply chain and vertical asset chain constraints – the essence of management's resource allocation challenge.

2.3 VPU Behavior

The logic employed to create value given a specific set of assets will vary greatly and depend in large measure on a VPU's value proposition and its associated computational requirements. However, a generalized model is instructive in understanding the timing properties of a real-time enterprise. The model used here is object-oriented and based on the concept of a *rational agent*. Its particular details are not critical to our timing arguments other than to demonstrate the various sources of variation in timeliness, and where resource management policies come to dominate the real-time nature of systems.

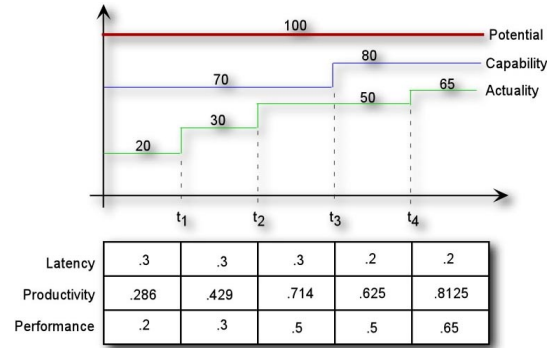


Figure 4 - Example VPU Performance Metrics

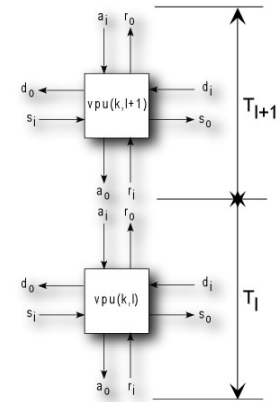


Figure 5a - Vertical (Asset Chain) Timing

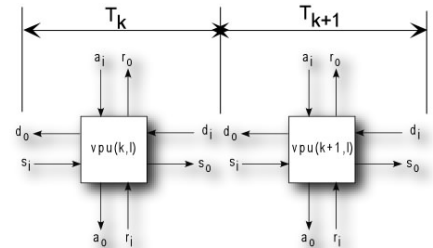


Figure 5b - Horizontal (Supply Chain) Timing

A VPU *object* has both state and behavior. A simplified state diagram for a rational VPU is given in figure 6. The states are defined as follows:

- s0: idle, suspended
- s1: unable, awaiting assets
- s2: able, awaiting a demand order
- s3: able, producing value
- s4: awaiting supply order fulfillment
- s5: awaiting subordinate returns
- s6: assessing capability, planning
- s7: demand fulfilled
- s8: error
- s9: reset

Timing variations can arise in each state, but the greatest potential variations are with states s1, s2, s4, and s5 where there are explicit [synchronous, blocking] waits for resources such as invested assets, customer orders, subordinate task completion, and material supplies.

Clearly, resource management (state s6) within the VPU is critical to its ability to maintain schedules, minimize delays and maximize throughput and yield. It is equally clear that a federation of VPUs, cascaded linearly or interconnected in an arbitrary mesh, may suffer deadlocks, long waiting queues, buffer overflows and other artifacts of chaotic or random behavior. Avoiding, or at least minimizing, such behavior is at the heart of the real-time enterprise management (control) problem, and is the source of the predictability requirement introduced earlier.

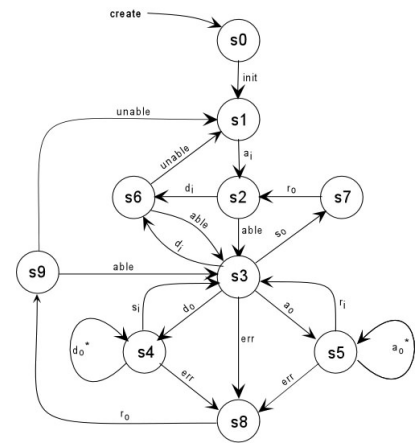


Figure 6 - VPU Rational Agent

A VPU's value to an enterprise may derive from a very low-level safety sensitive manufacturing activity with millisecond timing requirements (e.g., a chemical batch reactor) or a very high-level economic risk sensitive activity with megasecond timing requirements (e.g., buying raw materials on a commodities exchange). As such, VPU timing requirements may vary from microseconds to megaseconds. In both cases, real-time behavior demands that these time constraints be met, and that a penalty be exacted for being late, or early, or failing altogether to meet them.

Figure 7 emphasizes the challenges of production scheduling and some obvious sources of timing variation. Several production processes are running in parallel. For the one visible, task t_1 initiates the business

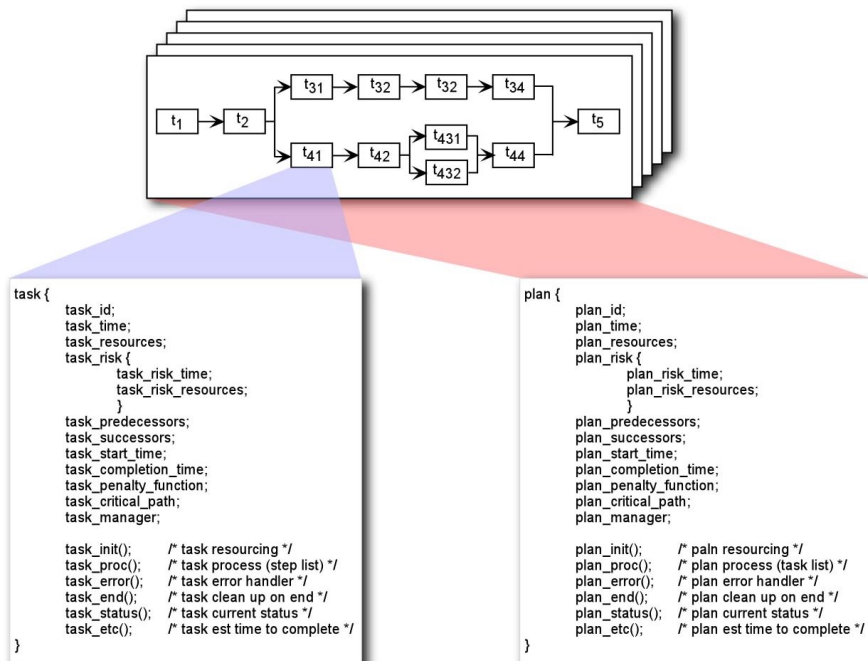


Figure 7 - VPU Business Processes: Resource Scheduling as a Source of Timing Variation

process and is followed by t_2 . After t_2 completes the process forks into two parallel activities denoted by t_3 and t_4 and their respective subtasks. These two threads rendezvous at t_5 where the process completes. The VPU responsible for executing the process depicted does so according to a state machine, such as in figure 6. Each task t_i may interact with other VPUs (i.e., customers or suppliers).

For each process there is a production plan with its associated (macro) timing requirements. Within a plan, the individual tasks contribute to its end-to-end performance. Individual variations in task completion times contribute to plan variations. Individual plan variations contribute to VPU variations, and so on throughout the metasystem. Important work has been done on grid-supported and economic-based distributed resource management [...], but the treatment of end-to-end timing remains a critically open issue [...]. Grid resource management remains non-real-time.

3 Value Transformations

The objective of a VPU is the creation of economic or information value. In general, it does so by transforming assets and policies it receives from its investors into capabilities to satisfy its customers' demands. These two objectives suggest that any model of a VPU will be characterized by two interdependent sets of transformations (re. figure 3). One set, the asset transformation functions, define the flow of assets down the asset chain with their respective time-delayed returns flowing upwards. The second set, the demand transformation functions, define the flow of customer demands to the left with the time-delayed fulfillment of those demands flowing to the right. Beginning with demand management we develop these two sets of transformations.

3.1 Demand Management Functions

The simplified demand transformation model depicted in figure 8 functions as follows. A customer order $\mathbf{d}_i(\mathbf{n})$ arrives at the VPU's demand port during the \mathbf{n}^{th} processing interval. The VPU's internal policies cause this demand to split (\mathbf{a}) into components that are to be produced from internal capabilities (\mathbf{d}_s), and those that require external orders $\mathbf{d}_o(\mathbf{n}+1)$ to its supply chain producers (\mathbf{d}_d). In the reverse direction, supply chain producers fulfill their previous orders $\mathbf{s}_i(\mathbf{n})$, which are split (\mathbf{b}) internally by policy into inventory stocks (\mathbf{s}_d) and components required to complete existing orders (\mathbf{s}_s), allowing the VPU to fulfill its demand $\mathbf{s}_o(\mathbf{n}+1)$. The time intervals represent processing times for the two flows, and include the end-to-end transport delays introduced by the grid.

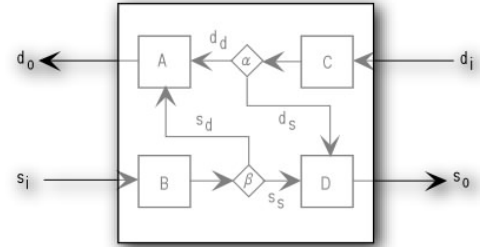


Figure 8 – VPU Demand Mgmt

The four functions (i.e., methods) labeled $\mathbf{A}[\mathbf{x}, \mathbf{y}]$, $\mathbf{B}[\mathbf{x}]$, $\mathbf{C}[\mathbf{x}]$, and $\mathbf{D}[\mathbf{x}, \mathbf{y}]$ represent the embodiment of the supply chain value propositions for the VPU, as executed by the state machine of figure 6. $\mathbf{A}[\mathbf{x}, \mathbf{y}]$ represents the command and control function responsible for placing demands on upstream suppliers, and $\mathbf{B}[\mathbf{x}]$ represents the command and control function responsible for accepting (e.g., paying for) demand completion. $\mathbf{C}[\mathbf{x}]$ accepts (commits to) customer demands, and $\mathbf{D}[\mathbf{x}, \mathbf{y}]$ fulfills that demand. Combined, they represent the VPU's supply chain value production transformations.

The demand transformation (i.e., production) equations are readily apparent and given by

$$\mathbf{d}_o(\mathbf{n}+1) := \mathbf{A}[\mathbf{d}_d(\mathbf{n}), \mathbf{s}_d(\mathbf{n})] = \mathbf{A}[\mathbf{a} * \mathbf{C}[\mathbf{d}_i(\mathbf{n})], (1-\mathbf{b}) * \mathbf{B}[\mathbf{s}_i(\mathbf{n})]] \quad (1)$$

$$\mathbf{s}_o(\mathbf{n}+1) := \mathbf{D}[\mathbf{d}_s(\mathbf{n}), \mathbf{s}_s(\mathbf{n})] = \mathbf{D}[(1-\mathbf{a}) * \mathbf{C}[\mathbf{d}_i(\mathbf{n})], \mathbf{b} * \mathbf{B}[\mathbf{s}_i(\mathbf{n})]] \quad (2)$$

Where the intermediate products are

$$\mathbf{d}_d(\mathbf{n}) := \mathbf{a} * \mathbf{C}[\mathbf{d}_i(\mathbf{n})] \quad (3)$$

$$\mathbf{d}_s(\mathbf{n}) := (1-\mathbf{a}) * \mathbf{C}[\mathbf{d}_i(\mathbf{n})] \quad (4)$$

$$\mathbf{s}_d(\mathbf{n}) := (1-\mathbf{b}) * \mathbf{B}[\mathbf{s}_i(\mathbf{n})] \quad (5)$$

$$\mathbf{s}_s(\mathbf{n}) := \mathbf{b} * \mathbf{B}[\mathbf{s}_i(\mathbf{n})] \quad (6)$$

3.1.1 Case 1: Stand-alone VPU

There are special cases of this general model worth noting. The first occurs when a VPU is self-sufficient and stand-alone. In this case, $a=1$, $b=0$, and $d_0(n) = s_i(n) = 0$, with the result that the production equations reduce to

$$s_o(n+1) := D [C [d_i(n)], 0] \quad (7)$$

3.1.2 Case 2: Serial Producers

The second special case occurs when a client VPU serially adds value to the production of a server VPU. In this case $a=1$ and $b=1$, yielding

$$d_o(n+1) := A [C [d_i(n)], B [s_i(n)]] \quad (8)$$

$$s_o(n+1) := D [C [d_i(n)], B [s_i(n)]] \quad (9)$$

3.2 Asset Management Functions

In a similar fashion (figure 9), the asset transformation (i.e., production) functions may be stated as

$$a_o(n+1) := F [a_a(n), r_a(n)] = F [d * E [a_i(n)], (1-g) * G [r_i(n)]] \quad (10)$$

$$r_o(n+1) := H [a_r(n), r_r(n)] = H [(1-d) * E [a_i(n)], g * G [r_i(n)]] \quad (11)$$

Here **E**, **F**, **G** and **H** are the embodiment of the asset chain value propositions for the VPU, again as executed by the state machine of figure 6. **E** [**x**] and **H** [**x**, **y**] represent the management functions responsible for generating returns on assets invested in this VPU. **F** [**x**, **y**] and **G** [**x**] represents the management functions responsible for investing assets in subordinate VPUs, and in leveraging their returns. The **d** and **g** controls perform policy-driven allocation functions similar to those discussed above.

For simplicity we have not indexed these equations to uniquely identify a particular VPU (k, l) or to distinguish the uniqueness of its various event times (n) or the length of time delays ($n+1$) of asset investment actions. Such requirements will be addressed subsequently.

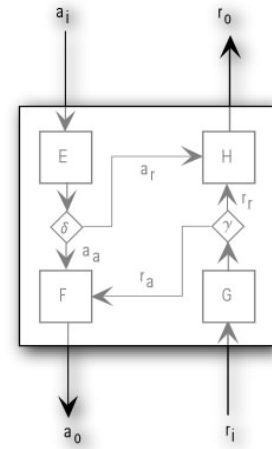


Figure 9 – VPU Asset Mgmt

3.2.1 Case 1: Stand-alone VPU

As in 3.1.1, the simplest case is a stand-alone VPU without subordinates requiring investments. In this case, $d=1$, $g=0$, yielding $a_o(n) = r_i(n) = 0$, with the result that the investment equations reduce to

$$r_o(n+1) := H [E [a_i(n)], 0] \quad (12)$$

3.2.2 Case 2: Serial Investors

As in 3.1.2, a VPU can exist solely as an asset manager, with a mission to invest in subordinate VPUs. In such cases investor VPUs are defined by $d=1$, $g=1$, and

$$a_o(n+1) := F [E [a_i(n)], G [r_i(n)]] \quad (13)$$

$$r_o(n+1) := H [E [a_i(n)], G [r_i(n)]] \quad (14)$$

3.3 Cascaded (Sequential) Demand Management

In order to develop the general cascaded (figures 3 and 5b) demand management equations we need to properly label (index) participating VPUs. To do so we add the *relative indexing* introduced in figure 3,

where $VPU[k, l]$ represents the k^{th} production at the l^{th} level of the enterprise hierarchy. This results in a more precise statement of equations (1) and (2).

$$d_o^{k,1}(n+1) := A^{k,1}[x^{k,1}, y^{k,1}] = A^{k,1}[d_d^{k,1}(n), s_d^{k,1}(n)] \quad (15)$$

$$s_o^{k,1}(n+1) := D^{k,1}[x^{k,1}, y^{k,1}] = D^{k,1}[d_s^{k,1}(n), s_s^{k,1}(n)] \quad (16)$$

Where the intermediate products are

$$d_d^{k,1}(n) := a^{k,1} * C^{k,1}[d_i^{k,1}(n)] \quad (17)$$

$$d_s^{k,1}(n) := (1-a^{k,1}) * C^{k,1}[d_i^{k,1}(n)] \quad (18)$$

$$s_d^{k,1}(n) := (1-b^{k,1}) * B^{k,1}[s_i^{k,1}(n)] \quad (19)$$

$$s_s^{k,1}(n) := b^{k,1} * B^{k,1}[s_i^{k,1}(n)] \quad (20)$$

The VPU demand interface boundary conditions are

$$d_i^{k-1,1}(n) := d_o^{k,1}(n) \quad (21)$$

$$d_o^{k+1,1}(n) := d_i^{k,1}(n) \quad (22)$$

$$s_o^{k-1,1}(n) := s_i^{k,1}(n) \quad (23)$$

$$s_i^{k+1,1}(n) := s_o^{k,1}(n) \quad (24)$$

Substitution, and eliminating some superscripts for clarity, yields the iterative version of the sequential demand management equations.

$$d_o^{k,1}(n+1) := A[a * C[d_o^{k+1,1}(n), (1-b) * B[s_o^{k-1,1}(n)]]] \quad (25)$$

$$s_o^{k,1}(n+1) := D[(1-a) * C[d_i^{k+1,1}(n)], b * B[s_i^{k-1,1}(n)]] \quad (26)$$

3.4 Cascaded (Sequential) Asset Management

As in 3.2 and 3.3, we note the VPU's asset interface boundary conditions

$$a_i^{k,1-1}(n) := a_o^{k,1}(n) \quad (27)$$

$$a_o^{k,1+1}(n) := a_i^{k,1}(n) \quad (28)$$

$$r_i^{k,1+1}(n) := r_o^{k,1}(n) \quad (29)$$

$$r_o^{k,1-1}(n) := r_i^{k,1}(n) \quad (30)$$

Substitution yields the iterated version of the sequential asset management equations

$$a_o^{k,1}(n+1) := F[d * E[a_o^{k,1+1}(n)], (1-g) * E[r_o^{k,1-1}(n)]] \quad (31)$$

$$r_o^{k,1}(n+1) := H[(1-a) * G[a_o^{k,1+1}(n)], g * G[r_o^{k,1-1}(n)]] \quad (32)$$

4 Computing Value Propositions

Value results from policies and mechanisms represented by the functions $A[x, y]$, $D[x, y]$, $F[x, y]$, and $H[x, y]$. There are many possible interpretations of these functions. In the *real* (non virtual) world, enterprises measure value creation through interpretation of operating data produced via transactions in various management applications subsystems and stored in various databases. Figure 10 summarizes the major applications found in today's corporations. The five rings represent the "1" dimension in the $VPU[k, l]$, where $l=0$ represents production devices (people and machines); $l=1$ represents production units; $l=2$ production areas; $l=3$ business units; $l=4$ business areas; and $l=5$ corporate enterprises. The radials are representative of areas of management command and control. The quadrants summarize major application domains. And finally, the colored overlay regions represent application suites provided by vendors of enterprise software subsystems.

We do not delve into the data and transaction services of these applications, but recognize their performance (re. Section 3.1) as a key contributor to the overall performance of any given VPU, especially if they are grid-connected and subject to performance anomalies resulting from end-to-end completion time uncertainties in servers and the Internet infrastructure. For the purposes of this exposition, we bundle these infrastructure uncertainties into the variations normally associated with applications identified in figure 10.

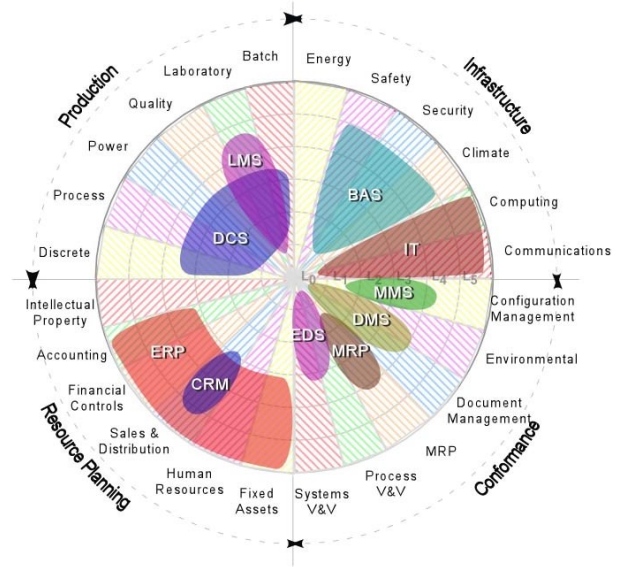


Figure 10 – Enterprise Application Systems

4.1 Performance Considerations

Figure 11 represents a distributed enterprise with six VPUs. VPU[2,3] serves equity market customers (investors) and provides assets and policies to its supply chain producers VPU[1,2], VPU[2,2] and VPU[3,2]. VPU[1,2], in turn, invests in its subordinate VPU[1,1] and VPU[2,2] invests in VPU[2,1].

The figure identifies supply chain timing intervals that define the enterprise’s ability to serve its product market customers. Demand $d_i^{3,2}(n)$ arrives at VPU[3,2] during interval “n.” VPU[3,2] issues demand $d_i^{2,2}(n+1)$ to its supplier VPU[2,2] during interval “n+1,” that results in a subsequent demand $d_i^{1,2}(n+2)$ to VPU[1,2]. In the return direction, demands are fulfilled, respectively, at $s_o^{1,2}(n+3)$, $s_o^{2,2}(n+4)$ and $s_o^{3,2}(n+5)$.

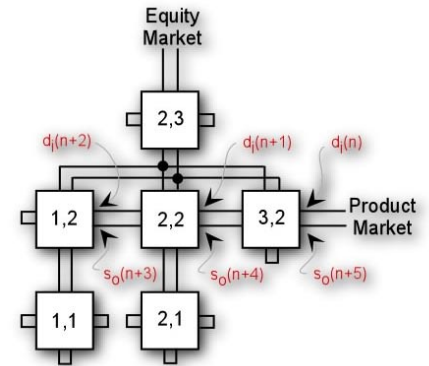


Figure 11 – VPU Timing

In this configuration, several command and control questions may be discussed. For example,

- ◆ What is the relationship between equity market returns and product market returns.
- ◆ What are VPU[2,3]’s alternative strategies for investing in its subordinate VPUs?
- ◆ What does VPU[3,2] need to do to allow VPU[2,3] to profitably lower its product market response time from n+5 to, say, n+4?
- ◆ What is the impact on return on assets at VPU[2,3] of increasing capacity of VPU[1,2]?

4.2 VPU Productivity

Enterprise performance may be represented by the relative productivity gains achieved by the transformation functions as assets and demand (orders) flow through the VPU. Here we define the relative gains as ratios of outputs to inputs for each of the ports on a VPU. For simplicity we have omitted the VPU indices [k,l], the function arguments, and time intervals. The details, for G_{aa} as an example, would be:

$$G_{aa}^{k,1}(n+1) := F^{k,1} [d^{k,1} * E [a_o^{k,1+1}(n)], (1-g^{k,1}) * E^{k,1} [r_o^{k,1-1}(n)]] / a_i^{k,1}(n)$$

	a_o	r_o	d_o	s_o
a_i	$G_{aa}=F[x, y] / a_i$	$G_{ra}=H[x, y] / a_i$	$G_{da}=A[x, y] / a_i$	$G_{sa}=D[x, y] / a_i$
r_i	$G_{ar}=F[x, y] / r_i$	$G_{rr}=H[x, y] / r_i$	$G_{dr}=A[x, y] / r_i$	$G_{sr}=D[x, y] / r_i$
d_i	$G_{ad}=F[x, y] / d_i$	$G_{rd}=H[x, y] / d_i$	$G_{dd}=A[x, y] / d_i$	$G_{sd}=D[x, y] / d_i$
s_i	$G_{as}=F[x, y] / s_i$	$G_{rs}=H[x, y] / s_i$	$G_{ds}=A[x, y] / s_i$	$G_{ss}=D[x, y] / s_i$

Table 1 – VPU Gain Equations

Highlighted cells are those typically viewed as critical by enterprise managements.

5 Examples

We are interested in the nature of enterprise command and control, especially in distributed real-time enterprise. The subject requires a concise model of enterprise, and of the interdependencies between and among extended enterprises – those made up of federations of semi-autonomous cooperating members. The VPU model given in the body of this paper provides a means of discussing enterprise behavior. We exercise this model here through three examples

5.1 Example 1 – Consolidated Supply Chain

Figure A1 describes an enterprise with three levels of asset management, as might be found in a business ($l=3$) with three plants ($l=2$), and two production lines ($l=1$). The example demonstrates a consolidated supply chain at level 2 among cooperating plants. Note that VPU[2,3] invests in, and expects returns from, all three plants, and can control internal transfer pricing.

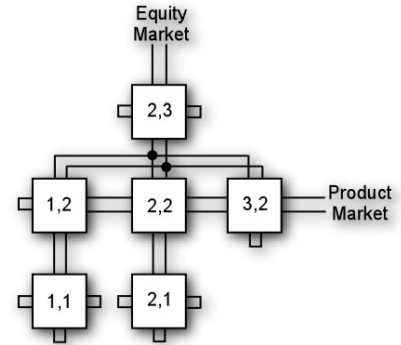


Figure A1 – Example 1

5.2 Example 2 – Parallel Product Market Servers

Figure A2 shows a single enterprise that simultaneously serves two product markets, one through VPU[3,3] and one through VPU[3,2]. In this configuration there are numerous practical questions about investment strategies in the two supply chains, about how investment returns from VPU[2,2] might be used to enhance capability in VPU[2,3], about investment returns from VPU[2,3] might be reinvested in upstream VPU[1,2] and downstream VPU[3,3] to improve performance in that chain, etc.

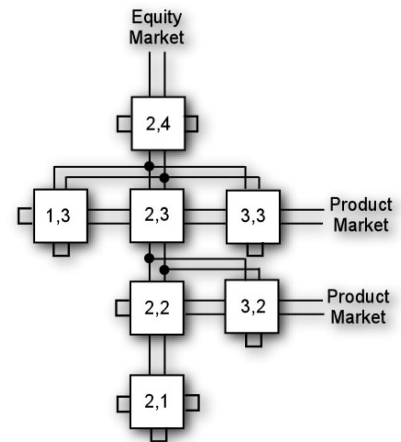


Figure A2 – Example 2

5.3 Example 3 – Parallel Equity Market Servers

Figure A3 is a simple case of two enterprises connected by VPU[4,3]. They each serve the equity market by providing goods and services to the same product market through the supply chain denoted by VPU[2,3], VPU[3,3], ... VPU[5,3]. In this configuration several important questions may be posed about the strategies and functioning of enterprise investment strategies in VPU[3,4] and VPU[5,4], policies in VPU[4,3] at the interface between the two enterprises, the efficacy of pricing and margins along the supply chain, and so on.

6 Conclusions

Enterprise, as a real-time computation of its value propositions, is a distributed computing system. Federated enterprises comprise interconnected webs of production systems whose collective behavior determines not only their own destiny, but that of the federation. In order to investigate the behavior of such federations, we have introduced the concept of a value production unit (VPU) and its computational role and behavior. The VPU model and its transformation functions provide the environment for modeling and simulating the performance of federations in various configurations. A subsequent paper will exercise such models.

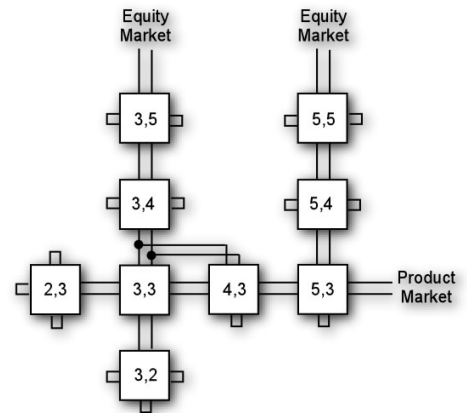


Figure A3 – Example 3

7 References

1. Albus, J., Outline for a Theory of Intelligence, IEEE Transactions on Systems, Man and Cybernetics, Vol 21, No 3, May-June 1991.
2. Antsaklis, P., Passino, K., Wang, S., *Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues*, Journal of Intelligent and Robotic Systems, 1:315-342, 1989.
3. Ashby, R., *Introduction to Cybernetics*, Chapman & Hall, Ltd, London, 1957, PDF version available by permission at www.echelon4.com.
4. Bayne, J., *Automation & Control in Large-scale Interactive Systems*, Proc IEEE International Symposium on Object-Oriented Real-time Computer Systems (ISORC), 2002.
5. Bayne, J., *Distributed Real-time Enterprise*, 2002, available at www.echelon4.com, a derivative of which has been submitted for publication to the IEEE Transactions on Systems, Man and Cybernetics.
6. Bayne, J., *Industrial Automation & Control in a Networked Economy*, invited talk, IEEE Workshop on Object-oriented Real-time Distributed Systems (WORDS), Santa Barbara, Ca., 1996.
7. Bayne, J., *Meta Systems*, ISA/TECH Conference Proceedings, Instrumentation, Systems and Automation (ISA) Society, 1996.
8. Bayne, J., *Scalability of Performance in High Volume Commercial Control Systems*, Proc IEEE International Symposium of Object-Oriented Real-time Systems (ISORC), 1999.
9. Bayne, J., Seem, J., Drees, K., *Adaptive Industrial Controls*, 7th International Symposium on Artificial Intelligence in Real-time Control (AIRC'98), October 1998, winner "Best Paper Award."
10. Bayne, J., *The Architecture of Enterprise*, an unpublished essay, available at www.echelon4.com, 2002.
11. Bayne, J., *The ELBA Object Model*, Ada Runtime Environment Working Group (ARTEWG), ACM SIG Ada, Santa Barbara, CA 1995.
12. Beer, S., *Brain of the Firm*, 2nd Ed., Wiley, 1994.
13. Buyya, R. and Murshed, M., *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), 1-32pp, Wiley Press, May 2002 (to appear).
14. Buyya, R., *Economic-based Distributed Resource Management and Scheduling for Grid Computing*, PhD Thesis, Monash University, Melbourne, Australia, April 12, 2002.
15. Buyya, R., et al, *Economic Models for Resource Management and Scheduling in Grid Computing, Special Issue on Grid Computing Environments*, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May, 2002 (to appear).
16. Chinnici, R., et al, *Web Services Definition Language (WSDL) - Version 1.2*, WC3 Working Group Draft, July 2002, www.w3.org/2002/07/wsdl.
17. Clark, R., Jensen, E., Wellings, A., Wells, D., *The Distributed Real-time Specification for Java: A Status Report*, www.real-time.org, Revised January 2002.
18. Copeland, T., Koller, T., Murrin, J., *Valuation – Measuring and Managing the Value of Companies*, 3rd Ed., Wiley, 2000.
19. De Roure, D., et al, *The Evolution of the Grid (pre print)*, Universities of Portsmouth and Southampton, UK. In Grid Computing: Making the Global Infrastructure a Reality.
20. Deering, S., Hinden, R., *Internet Protocol, Version 6 (IPv6) Specification*, Network Working Group, Internet Engineering Task Force (IETF), December 1998, <ftp://ftp.isi.edu/in-notes/rfc2460.txt>
21. DOD, *C4ISR Architecture Framework - Version 2.0*, December 1997.
22. DOD, *E-6 Command, Control, Communications, & Intelligence ACTD Proposals*, September 2001.
23. DOD, *Joint Tactical Architecture – Version 4.0*, April 2001.
24. Foster, I., et al, *Grid Services for Distributed System Integration*, IEEE Computer, June 2002.
25. Foster, I., et al, *The Physiology of the Grid – An Open Grid Services Architecture for Distributed Systems Integration (DRAFT)*, Argonne National Lab, June 2002.

26. Foster, I., et al, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft 2.9)*, June 2002.
27. Foster, I., *The Grid: A New Infrastructure for 21st Century Science (Reprint)*, Physics Today <http://www.aip.org/pt/vol-55/iss-2/p42.html>
28. Halang, W. and Stoyenko, A., Ed., Real Time Computing, NATO ASI Series F: Computer and Systems Sciences, Vol 127, 1991.
29. Harry, M. and Schroeder, R., Six Sigma, Currency (Doubleday), 2000.
30. Henoch, J. and Ulrich, H., *Agent-based Simulation Platform for Evaluating Management Concepts*, Proceedings of EUROSIM 2001, June 2001
31. Jaques, E., Requisite Organization – The CEO’s Guide to Creative Structure & Leadership, Cason Hall, 1989.
32. Jensen, E., Locke, C., and Tokuda, H., *A Time-Value Driven Scheduling Model for Real-time Operating Systems*, Proc. Symposium on Real-time Systems, IEEE, November 1985
33. Jensen, E., *The Distributed Real-Time Specification for Java – An Initial Proposal*, Journal of Computer Systems Science & Engineering, March 2001.
34. Jensen, E., *Utility Functions: A General Scalable Technology for Software Execution Timeliness as a Quality of Service*, Proc. Software Technology Conf., Utah State Univ., April 2000.
35. Johnson, W. and Brooke, J., *Core Grid Functions: A Minimal Architecture for Grids*, Working Draft 3, Grid Protocol Architecture Working Group, Global Grid Forum, July 2002.
36. Kishimoto, H. and Snelling, D., *OSGA Fundamental Services: Requirements for Commercial Grid Systems*, OSGA Working Group, Global Grid Forum, October 2002.
37. Lessig, L., *Code, and Other Laws of Cyberspace*, Basic Books, 1999. See also www.cyberlaw.stanford.edu/lessig/.
38. McKinsey, J., Introduction to the Theory of Games, McGraw-Hill, 1952.
39. Pouchard, L., et al, *Ontology Engineering for Distributed Collaboration in Manufacturing*, Proceedings of the AIS2000 Conference, March 2000.
40. Rajic, H. et al, *Distributed Resource Management Application API Specification*, Global Grid Forum, September 2002.
41. Schlenoff, C., et al., *The Process Specification Language (PSL), Overview and Version 1.0 Specification*, NISTIR 6459, National Institute of Standards and Technology, Gaithersburg, MD (2000).
42. Schwiegelshohn, U. and Yahyapour, R., *Grid Scheduling Architecture*, Working Draft 1, Global Grid Forum, October 2002.
43. Scrudder, R., Lutz, R., Dahmann, J., *Automation of HLA Federation Development and Execution Process*, williams.af.mil/ambr3.htm.
44. Selig, B., Gullekson, G., Ward, P., Real-time Object-Oriented Modeling, Wiley, 1994.
45. Smith, W., *A Framework for Control and Observation in Distributed Environments*, NAS Technical Report #NAS-01-006, June 2001.
46. Sun Microsystems, *Java Remote Method Invocation Specification*, December 1999.
47. Wooldridge, M., Reasoning About Rational Agents, MIT Press, 2000.