GRID MODERNIZATION LABORATORY CONSORTIUM
U.S. Department of Energy

A Mathematical Representation of System Architectures

Application to Grid Architecture

March 2018

JD Taft

PNNL-27387

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <http://www.ntis.gov/about/form.aspx>
Online ordering: http://www.ntis.gov

This document was printed on recycled paper.

(8/2010)

A Mathematical Representation of System Architectures

Application to Grid Architecture

JD Taft[1]

March 2018

[1] Chief Architect for Electric Grid Transformation, Pacific Northwest National Laboratory

Contents

Figures

1.0 Background/Purpose

Much architecture work is ad hoc in nature, leading to difficulties in comparing architectural alternatives and determining how well a given proposed architecture fits the purposes for which it was devised. The lack of underlying rigor in the representation of architectures in terms of how elements contribute (or do not) to the goals of the architecture limits our ability to be analytical about architecture. The purpose of this work is to provide an approach to treating system architectures analytically, with the eventual goal of being able to quantify objectively the goodness of fit of system architectures, to be able to compare architectures rigorously, and to be able to optimize architectures with respect to specified goals.

2.0 Overview

The approach taken in this work is to represent architectural characteristics in a formal abstract space framework, and then to use graphs to map architectural elements to the characteristics, to convert the graphs to matrix representations, and finally to use matrix algebra, graph theory, and the theory of abstract spaces to develop both norms and metrics for system architectures. The abilities to evaluate, compare, and optimize architectures will follow from these essentials.

By mapping architectural elements in the manner described above, we hope to make it possible to draw upon the vast bodies of knowledge already available to us, in particular the matrix methods. With that in mind, we will drive the formulation in the direction of such tools, even when it might appear that other approaches could serve. Ultimately, we wish to encapsulate much of what follows into software tools so that practicing architects and others might be able to use the methods without having to master arcane mathematics, much in the manner one might use a financial spreadsheet without having to manipulate all of the financial equations directly.

3.0 Architecture Definition

An architecture is an abstract depiction of a system, consisting of <u>black box components</u>, <u>structure</u>, and <u>externally visible characteristics</u>. An architecture is not a design, nor is it just about interoperability. It is the highest level representation of a complex system and its purposes are several:

- Enable reasoning about a system’s structure and behavior
- Enable prediction of system characteristics
- Manage system complexity
- Facilitate communication among stakeholders (internal and external)
- Manifest earliest design decisions/constraints
- Identify gaps in theory, technology, organization, regulation…
- Identify/define interfaces and platforms

Structure provides the framework within which components combine to yield systems that deliver specific capabilities (see Figure 1).

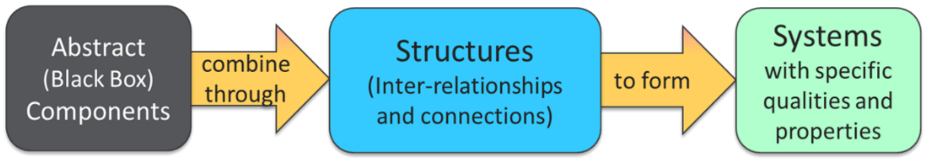
Abstract
(Black Box)
Components
combine through
Structures
(Inter-relationships and connections)
to form
Systems
with specific qualities and properties

Figure 1. Architectural Elements and Systems

4.0 Two Classes of Characteristics

Consider the nature of the characteristics by which we can describe a system architecture. There is much in the literature about "ility" words (reliability, flexibility, etc.), and such terms are often bandied about when the purposes and values of system architectures are discussed. Unfortunately, most of these terms are not rigorously and unambiguously defined, and do not have mathematical bases that would make them susceptible to analytical treatment. There is a second, more subtle problem: there are really two sets of characteristics to consider and they come from two different points of view. Mixing these indiscriminately (a common situation) leads to confusion that inhibits the ability to understand and compare architectures.

We divide the characteristics of a system architecture into two sets, arbitrarily labeled **qualities** and **properties**.

Qualities are those characteristics seen by the users of an architecture (more properly: the system that implements the architecture) and are closely related to the value of the architecture and therefore its goodness of fit. One may think of qualities as high level requirements expressed qualitatively or quantitatively.

Properties are those characteristics seen by the developers, builders, and operators of a system represented by the system architecture that are needed to deliver the qualities desired by the users. We recognize that system properties emerge from two attributes: intrinsic characteristics and functional capabilities. System intrinsic characteristics are mostly associated with structure whereas system functional capabilities are mostly associated with components. System properties emerge from structure and components and are what enable the system qualities to be manifested. Figure 2 below illustrates the concept.

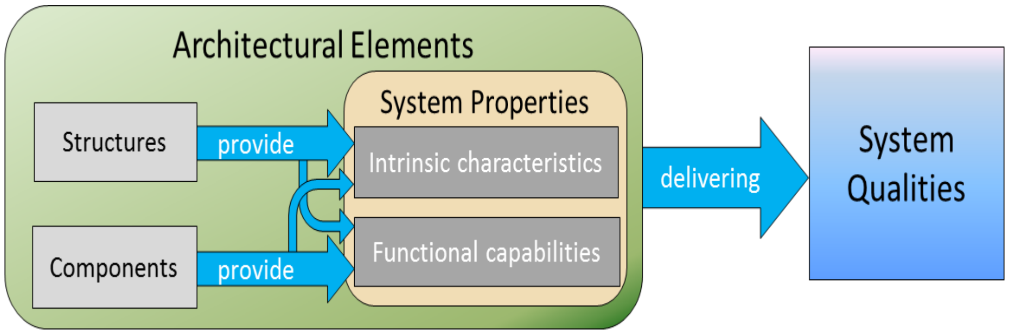

Figure 2. Architecture Analysis/Evaluation

Many of the intrinsic characteristics of a system are determined by its structures (in particular key constraints on system capabilities) and functional capabilities are largely determined by the components.

As the diagram illustrates, this is not a strict separation; hence the cross coupling.[1] This diagram is the beginning of the general process of mapping architectural elements in such a way that they can be treated analytically.

In the synthesis of architecture, we start with the user needs and follow the flow illustrated in Figure 3. From this standpoint, we can view the qualities as residing in the problem domain, and the properties as residing in the solution domain.

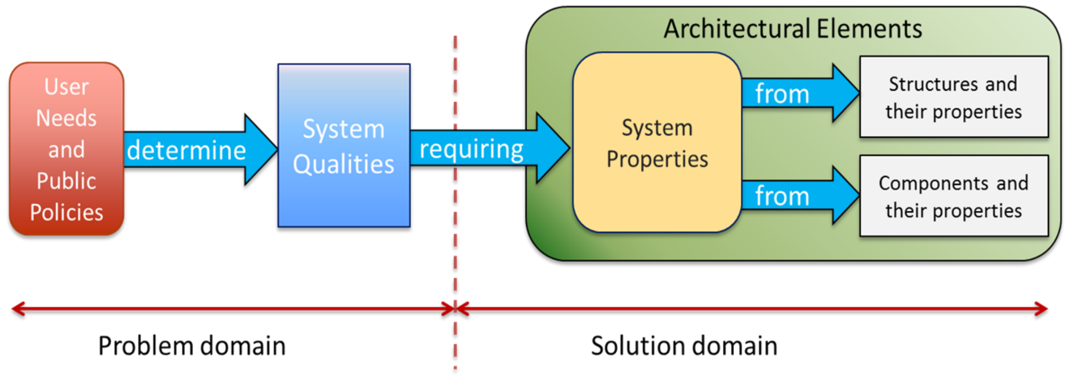

Figure 3. Architecture Synthesis

For the remainder of this paper we will focus on the analysis of architectures, and thus on the model of Figure 2. It will be convenient however to work backwards from qualities to architectural elements in the process of developing the representation schema.

5.0 Mathematical Preliminaries

In what follows, we will make use of a number of mathematical notions and while we will keep the complexity to a minimum, some use of advanced concepts and notations will be necessary. This section outlines the tools we will employ.

5.1 Graphs

A graph G(V,E) is a structure which consists of a set of vertices V = {v1, v2, …} and a set of edges E = {e1, e2,…}. Each edge connects or *is incident to* an unordered pair of (not necessarily distinct) vertices.[2] A wide variety of graph types and graph characteristics have been defined, and the mathematics of graph theory is extensive.

Graphs are often depicted diagrammatically, such as in the example of Figure 4, with vertices shown as circles or dots, and edges as straight or curved lines. Vertices and edges may be labeled and edges may have values and/or directionality associated with them. A graph for which the edges have directionality (indicated by an arrowhead), the graph is referred to as a directed graph or digraph.

[1] For this reason, the concepts of form and function do not have simple 1:1 mappings with structure and components.
[2] Shimon Even, Graph Algorithms, Computer Science Press, 1979.

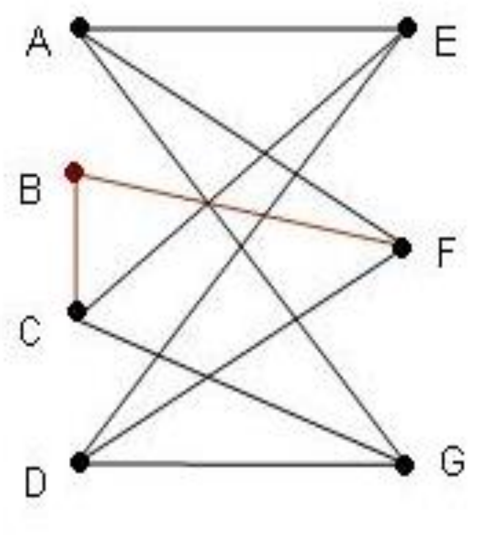

Figure 4. Example Graph

A k-partite graph is a graph whose vertices are or can be partitioned into k different independent sets. For k = 3, the graph is called a tripartite graph.

Various representations of graphs have been developed for purposes of computer processing, including incidence lists and adjacency matrices.

5.2 Topological Structure, Norms and Metrics, and Orthogonality

We shall employ a number of concepts from the theory of abstract spaces.[3] Here we list the key definitions very briefly.

A metric space is a nonempty set X and a function that maps X to the real line. The mapping must satisfy four axioms that define the essence of a metric, including non-negativity, reflexivity, and the triangle inequality. In essence, a metric is a measure of distance.

A linear space is a nonempty set X and two operations defined on the set: addition and scalar multiplication. These operations must satisfy seven conditions, including existence of elements 0 and 1, associativity, commutativity, existence of unique negatives for every element of X, and superposition.

Norms are functions on a linear space that satisfy four conditions, including non-negativity, scaling, and the triangle inequality for all elements of X. A norm is essentially a measure of size and can be used to induce a metric by applying other properties of linear spaces.

An inner product space is a real or complex linear space with an inner product - a function on pairs of elements of X that maps to the real line and that satisfies four conditions: distribution, scaling, non-negativity, and conjugate equality. For such a space, the norm of an element of X is equal to the square

[3] For a more complete review discussion, see Andrew P. Sage and Chelsea C. White III, Optimum Systems Control, Prentice Hall, 1977, pp. 387-394.

root of the inner product of an element with itself. A complete[4] inner product space is called a Hilbert space.

Two elements x and y of an inner product space are said to be orthogonal if their inner product is zero. Orthogonality corresponds to the geometric concept of perpendicularity.

5.3 Vectors, Matrices, and Matrix Notation

The algebra, calculus, and differential equations of vectors and matrices are very well established. We will make use of this body of knowledge and here we provide some basic notation definitions.

- A set of mn real or complex scalar quantities arranged in a rectangular array of n columns and m rows is a matrix of order (m, n) or m x n.
- A set of n quantities arranged in a single column of a matrix of order (n, 1) is a (column) vector.
- A set of n quantities arranged in a single row of a matrix of order (1, n) is a (row) vector.
- Transposition is the operation of interchanging rows and columns of a matrix. A column vector can be converted to a row vector by transposition.
- A matrix inverse is a matrix such that the product of the matrix and its inverse is in identity matrix.

Notation:

- **F** – matrix (bold capital letter)
- $\mathbf{F}^T$ – matrix transpose (superscript capital T denotes transpose)
- $\mathbf{F}^{-1}$ – matrix inverse (superscript -1 denotes inverse)
- **x** – column vector (bold lower case letter)
- $\mathbf{x}^T$ – row vector (transpose of a column vector)
- a or A – scalar quantity (non-bold letter)
- diag(**x**) – square matrix constructed by making the elements of **x** the diagonal elements of the matrix; all other entries in the matrix are zero
- $\mathbf{1}_n$ – column vector of order n where every entry is equal to 1
- $\mathbf{I}_n$ – identity matrix of order n x n (main diagonal entries equal to 1; all other entries equal to 0)

6.0 Architecture Representation

In this section we will develop the representation of system architectures in a Hilbert space. The process involves creating a tripartite graph, mapping elements into the graph, and then converting the graph to a linear space representation. The development of norms and metrics will proceed from there.

[4] A metric space is said to be complete if each Cauchy sequence in it is a convergent sequence. A Banach space is a complete (under the metric induced by the norm) normed space.

6.1 Architecture Graphs and Mappings

We begin by creating two levels of mappings: properties to qualities and architectural elements to properties. Here we presume that a set of system qualities and a set of system properties have been selected. The choice of qualities and properties is dependent on the type of system and the problem domain involved. This section will focus on the mechanics of developing representations; a later section will consider the use of this method for electric power grids and will discuss qualities and properties for grids.

Map the architecture elements, properties, and qualities in the form of a tripartite directed graph. In general, these graphs are not complete. The mappings will be indicated by directed edges, where the edges have values associated with them that represent the contribution to or level of support for the target property or quality. Hence the architectural elements provide support for one or more properties; the properties provide support for one or more qualities. The presence of an edge indicated support; the value associated with the edge indicates the magnitude of the support or contribution.

Step 1: Set up the nodes for the tripartite graph

Start by drawing the architectural elements, properties and qualities as three sets of nodes, as shown in Figure 5. In practice, the nodes may be boxes with labels naming the individual elements, properties and qualities.

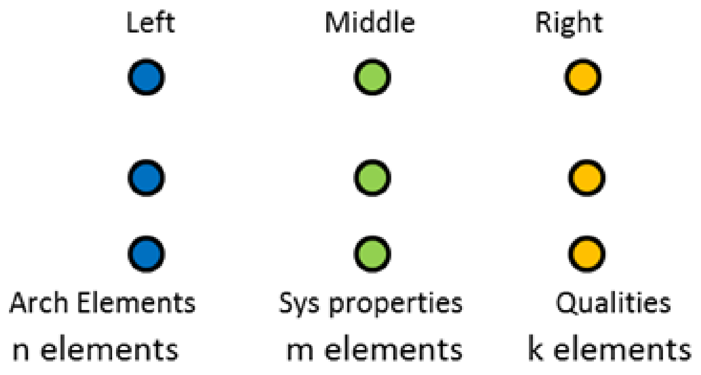

Figure 5. Example Tripartite Architecture Graph Nodes

In most cases, n>> m > k.

Step 2: Map properties to qualities

Proceed by mapping the properties (middle column) to the qualities as shown for in Figure 6. For each case where a property supports of contributes to a quality, draw an arrow from property to quality. Continue until all contributions are mapped. For the moment, the values of the mapping edges will be treated as binary, e.g. if an edge is present, its value of one; if not, its value is zero. Non-binary valuations are discussed below.

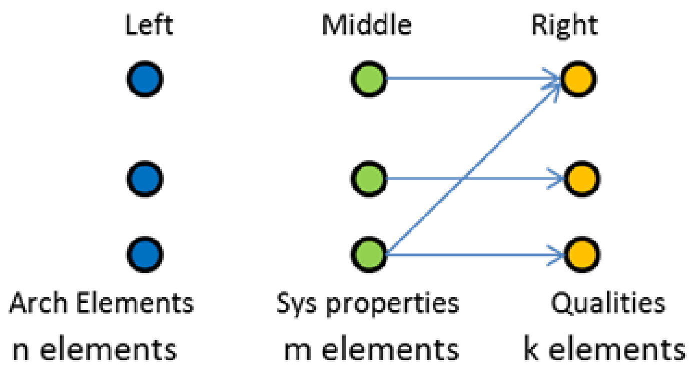

Figure 6. Example Properties to Qualities Mapping

Step 3: Map architectural elements to properties

Finally, map the architectural elements to the system properties to finish the graph as shown below in Figure 7. As with the previous part, for each case where an element supports of contributes to a property, draw an arrow from element to property.

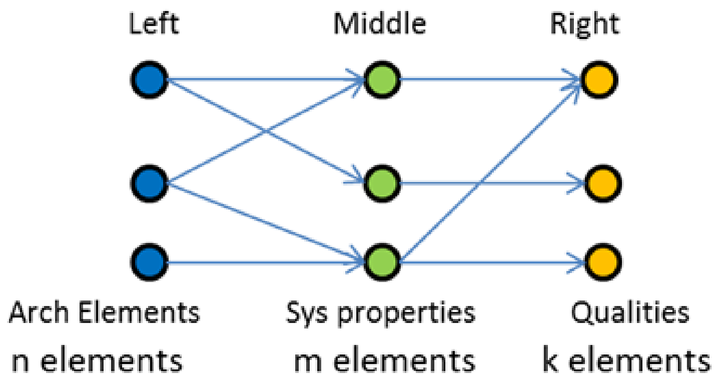

Figure 7. Example Completed Mapping

Generally, the number of architectural elements will be very much larger than the number or properties and it may be convenient to construct the graph in two sections – once with just the architectural components and once with the architectural structures. Ultimately these will be merged

The reason for doing the mapping in two stages (properties → qualities and elements → properties) is that it is generally very difficult to perform the mapping from elements to qualities in one step, especially when the links have non-binary values. Even with binary mappings, a one-step mapping is difficult. This is because the properties and qualities exist in different domains, as described above and this creates confusion about the connections between elements and characteristics. The two stage mapping provides structure and order to the mapping process that makes this problem manageable (keep in mind that real problems are much more complex than these simple diagrams make it appear). A slightly more realistic example using just architectural structures is shown in Figure 8.

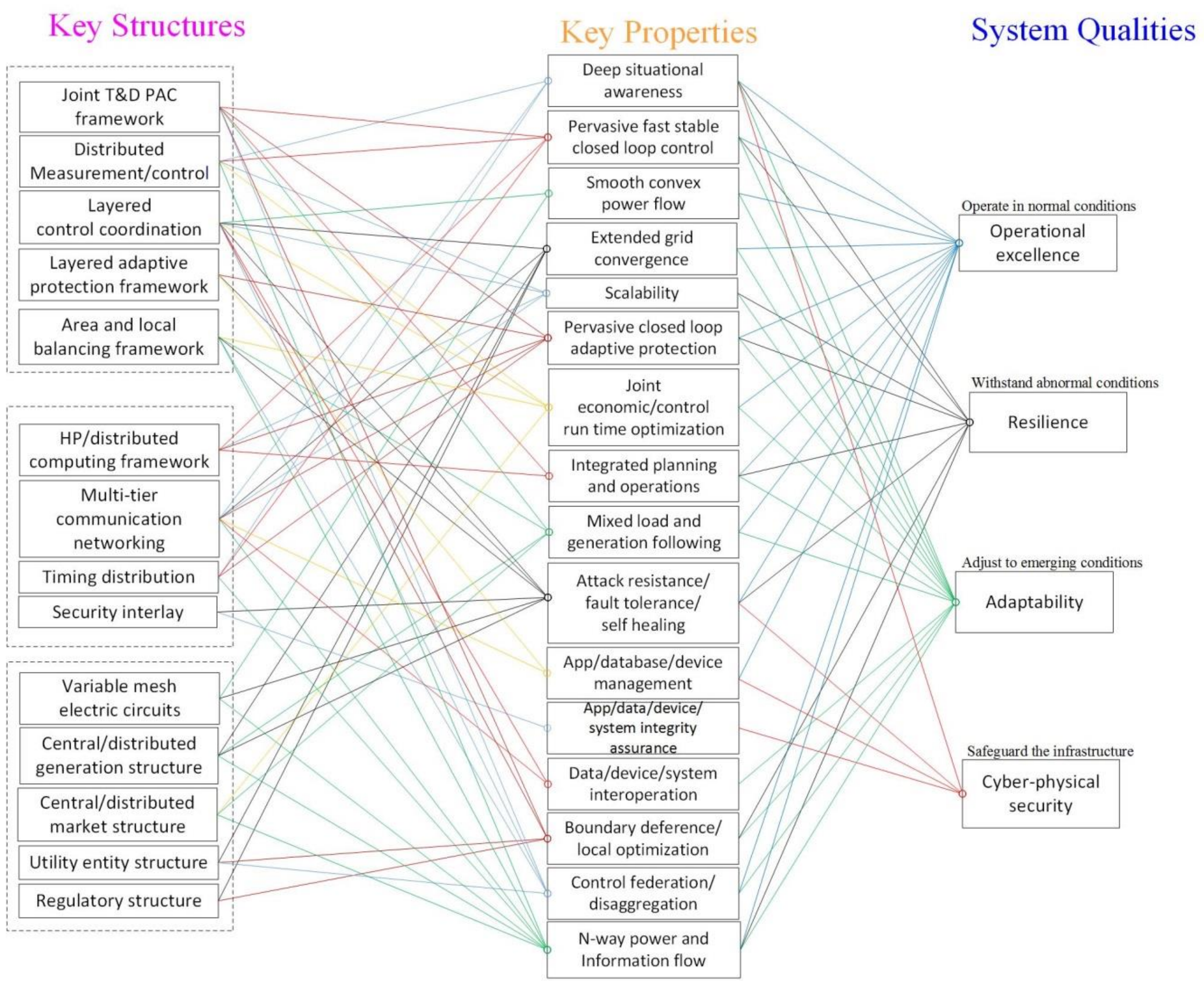
Key Structures
Key Properties
System Qualities
Joint T&D PAC framework
Distributed Measurement/control
Layered control coordination
Layered adaptive protection framework
Area and local balancing framework
HP/distributed computing framework
Multi-tier communication networking
Timing distribution
Security interlay
Variable mesh electric circuits
Central/distributed generation structure
Central/distributed market structure
Utility entity structure
Regulatory structure
Deep situational awareness
Pervasive fast stable closed loop control
Smooth convex power flow
Extended grid convergence
Scalability
Pervasive closed loop adaptive protection
Joint economic/control run time optimization
Integrated planning and operations
Mixed load and generation following
Attack resistance/ fault tolerance/ self healing
App/database/device management
App/data/device/ system integrity assurance
Data/device/system interoperation
Boundary deference/ local optimization
Control federation/ disaggregation
N-way power and Information flow
Operate in normal conditions
Operational excellence
Withstand abnormal conditions
Resilience
Adjust to emerging conditions
Adaptability
Safeguard the infrastructure
Cyber-physical security

Figure 8. A More Realistic Mapping Example

In practice, the mapping from properties to qualities will generally be static, but the mapping from architectural elements to properties will change as the architecture is developed. In fact, when comparing architectures as described below, it will be necessary to keep the property-to-quality mapping constant across architectural alternatives.

While a first approximation to the mappings can be made with binary-values links, a more nuanced mapping is possible by relaxing this constraint to allow for non-negative real values for links. The values are non-negative because the arrows represent *contribution or support*. The issue of how an element may *detract* from a property or quality is handled later.

The determination of non-binary link values is difficult in general. Three primary methods are available:

1. Use of subject matter expertise, engineering studies, and empirical data to place values on the contributions
2. Use of simulation studies to determine relative contributions
3. Use of graph-theoretic methods and measures

This last approach is developmental and is based on introducing measures from graph theory and algebraic topology into the network structure representations of architectural structures and does not address contributions by components. Fortunately, component contributions are more easily handled by the first two methods. Later we shall explore a use for the binary version of the **L** matrix in terms of architecture complexity.

6.2 Converting Architecture Graphs to Matrices

In the next section, we show how to transform the representation to the domain of linear spaces to develop the Architecture Equation. We begin by converting the two sets of mappings to matrices. In order to do this we define three vectors:

- **q** – the k x 1 vector of qualities
- **p** – the m x 1 vector of properties
- **e** – the n x 1 vector of architectural elements

For the elements vector **e**, each entry indicated the presence of the corresponding architectural element with a one or its absence with a zero. As we shall see later, we may turn architectural elements “on” and “off” (include or exclude them) by setting the value of the corresponding element of **e**.

Consider the mapping from elements to properties as one transition and the mapping from properties to qualities as a second transition, and define a mapping matrix for each. Hence we define the left mapping matrix **L** as an m x n real matrix such that

$$\mathbf{p} = \mathbf{L}\ \mathbf{e}.$$

The **L** matrix may be assembled from the left portion of the graph by setting matrix entry (i,j) to the value of the link connecting element i (1 <= i <= n) with property j (1 <= j <= m). If no link exists for any (i,j), the corresponding **L** matrix entry is set to zero. Thus, for the example of Figure 7,

$$\mathbf{L} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Thus the elements of the property vector attain values by summing architectural elements into them. The **L** matrix is the *architectural element to property mapping*. This matrix may vary with alternative architectures for the same problem since the architectural elements may vary.

Next we define the right mapping matrix **R** as a k x m real matrix such that

$$\mathbf{q} = \mathbf{R}\ \mathbf{p}.$$

The **R** matrix may be assembled from the right portion of the graph by setting **R** matrix entry (i,j) to the value of the link connecting property i (1 <= i <= m) with quality j (1 <= j <= k). If no link exists for any (i,j), the corresponding matrix entry is set to zero. Thus, for the example of Figure 7,

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The **R** matrix is the *property to quality mapping*. This matrix is normally constant for a given architectural problem.

Finally, we may compute the basic mapping from architectural elements to qualities as

$$\mathbf{q} = \mathbf{R}\,\mathbf{L}\mathbf{e}.$$

6.3 Architectural Detractors

Some architectural elements may detract from certain properties or from certain qualities instead of contributing to them. Most notably, some elements that add capability can also reduce reliability. Rather than putting these detractors into the graphs, we model them by introducing detraction matrices directly to map these effects. We define the property detraction matrix $\mathbf{D}_p$ as a matrix with the same dimensions as **L** that obeys the following constraints:

$$\mathbf{D}_p(i,j) = 0 \quad \text{if } \mathbf{L}(i,j) \neq 0$$

$$\mathbf{D}_p \geq 0$$

So, elements of $\mathbf{D}_p$ are non-negative and are zero if the corresponding link in **L** exists (is non-zero). This constraint exists so that elements of **L** represent gross contribution. We then introduce the property detractor into the mapping equation as

$$\mathbf{q} = \mathbf{R}\,(\mathbf{L} - \mathbf{D}_p)\,\mathbf{e}.$$

Quality detraction can be modeled in a similar fashion. We define quality detractor matrix $\mathbf{D}_q$ with the same dimensions as the matrix product **RL** (i.e. k x n). Introducing the quality detractor into the equation for **q** yields

$$\mathbf{q} = [\,\mathbf{R}\,(\mathbf{L} - \mathbf{D}_p) - \mathbf{D}_q\,]\;\mathbf{e}.$$

This then is the architecture equation, and we may compute the total mapping matrix **T** as

$$\mathbf{T} = \mathbf{R}\,(\mathbf{L} - \mathbf{D}_p) - \mathbf{D}_q.$$

The **T** matrix is the complete mapping from architectural elements to qualities. We can envision the computational flow as shown in Figure 9 below.

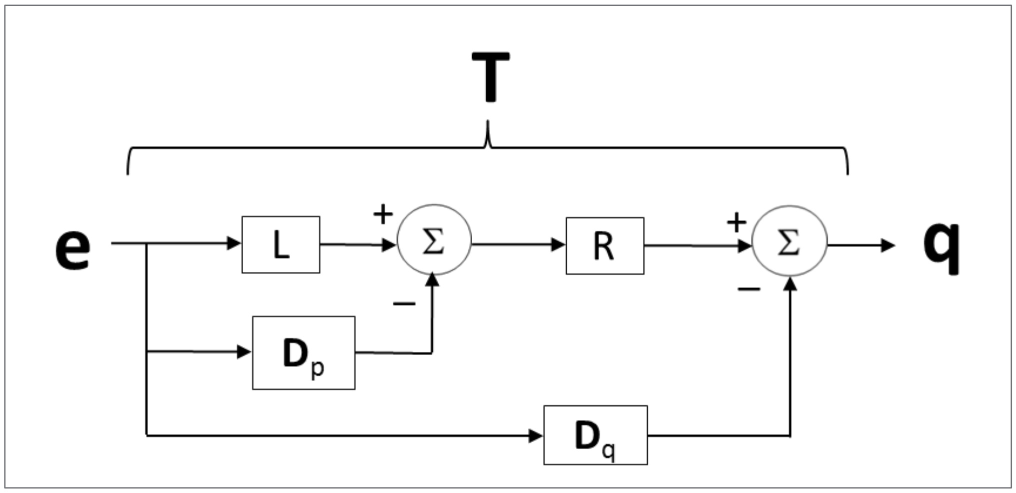

Figure 9. Architectural Mapping Computation Flow

6.4 Qualities as Basis Vectors

Consider the set of qualities as a set of orthonormal basis vectors in a Hilbert space. The **T** matrix can therefore be considered to be the projection of the architecture onto this set of basis vectors (the qualities) in a complete inner product space. For this to work, the qualities must be defined in a way that is mathematically compatible with the foregoing statement which is why we have needed the formalisms introduced above.[5] We shall designate the basis set of orthonormal quality vectors as $\mathbf{q}_0$.

We may view **q** as being <u>the architecture vector</u> for any particular architecture. Now let us introduce an architecture reference vector by weighting the quality basis vectors. We define a set $\{w_i\}$ of k values to be relative weightings for the qualities, that is, each w_i is a non-negative real value that expresses the relative importance of the corresponding quality to the stakeholders. The set of w_i must satisfy the constraints

$$w_i \geq 0; \quad 1 \leq i \leq k$$

$$\sum w_i = 1.0.$$

We can now define the normalized architectural reference vector **r** by making the elements of $\{w_i\}$ the main diagonal entries of a matrix **W** whose other entries are all zero. The reference vector **r** is then

$$\mathbf{r} = \mathbf{W}\mathbf{q}_0 / \| \mathbf{W}\mathbf{q}_0 \|$$

The reference vector represents the idealized architectural direction. We may now introduce a norm and a metric for architectures.[6] First we introduce the architectural *alignment metric* as the real scalar quantity

$$A(\mathbf{q}, \mathbf{r}) = < \mathbf{q}, \mathbf{r} > /[\ \|\mathbf{q}\|\ \|\mathbf{r}\|\ ],$$

[5] We hypothesize but have not proven that if the qualities are orthogonal, the properties do not have to be.
[6] Here we are *not* using the norm to induce the metric. They are both intended for characterization of architectures.

where < , > is the inner product of vectors and $||\mathbf{x}||$ is the ℓ^2 norm[7] of the vector $\mathbf{x}$. This metric measures the distance between the architectural reference vector and the specific architecture vector. We may calculate the distance between two architectures $\mathbf{q}_1$ and $\mathbf{q}_2$ in the same manner as

$$A(\mathbf{q}_1,\mathbf{q}_2) = <\mathbf{q}_1 , \mathbf{q}_2> / (||\mathbf{q}_1|| \ ||\mathbf{q}_2||)$$

The inverse cosine of A is therefore the angle between two architectures.

Second, we introduce the intuitive concept of architectural *strength*, which is a measure of how aggressively an architecture supports the architectural reference. The strength of $\mathbf{q}$ with respect to $\mathbf{r}$ is a real scalar defined as

$$S(\mathbf{q} , \mathbf{r}) = < \mathbf{q} , \mathbf{r} > / \mathbf{e}^T\mathbf{e}.$$

That is to say, strength is the magnitude of the projection of the architecture vector onto the reference vector, divided by the number of architectural elements employed. The reason for dividing by the number of elements is to keep the strength norm from growing just due to the total number of architectural elements. This combats a tendency to add complexity for its own sake.

The combination of the alignment metric and the strength norm may be thought of as indicating the effectiveness of the architecture in meeting the requirements of the stakeholders, as reflected in the quality and property mappings.

7.0 Architecture Analysis and Optimization

Given the graph and matrix representation schema for system architectures, we may consider how to further characterize architectures and how to perform differential analysis and optimization of architectures.

7.1 Meta-Structure Analysis and Complexity

We may designate the 5-tuple [$\mathbf{e}$, $\mathbf{L}$, $\mathbf{R}$, $\mathbf{D}_p$, $\mathbf{D}_q$] as an architecture, for mathematical purposes. Given this matrix representation of a system architecture, we examine the structure of the matrices to gain understanding about the underlying architecture – a process just starting to be developed that we refer to as architectural meta-structure analysis.

We can compute the following, using the binary version[8] of the $\mathbf{L}$ matrix:

- Number of architectural elements: $\mathbf{e}^T\mathbf{e}$
- Number of links (graph edges) in $\mathbf{L}$: trace ($\mathbf{L}^T\mathbf{L}$) or trace ($\mathbf{L}\mathbf{L}^T$)
- Maximum element fan-out (number of links exiting the element): $||\mathbf{L}||_1$ (max column sum)
- Largest property fan-in (links connecting into a property): $||\mathbf{L}^T||_1$ (max row sum)

We may consider this set of values as characterizations of the *complexity* of the architecture. [9]

[7] Also known as Euclidean norm; intuitively, the length of the vector.
[8] Entries are either one or zero.
[9] Other analyses of the matrices may yield further structural insights but have not yet been pursued.

7.2 Differential Analysis of Architectures

Two differential analysis problems present themselves:

- Determining incremental impact of individual architectural elements or element sub-groups
- Comparing alternative architectures

Both problems are facilitated by the availability of the alignment metric and the strength norm. For the element/sub-group differential analysis, one may simply set selected elements of the **e** vector to zero to "turn off" those elements and then compare the resultant values of A and S to those of the baseline case. The **L** and **R** matrices are kept constant during this exercise, as are the $\mathbf{D}_p$ and $\mathbf{D}_q$ matrices.

Comparing architectures is a matter of doing the same for entire architecture. This means that while the R matrix will be constant, the **e** vector and the **L** matrix would likely change for each proposed architecture. The $\mathbf{D}_p$ and $\mathbf{D}_q$ matrices may change as well.

7.3 Architecture Optimization

Once differential comparison are possible, then we may consider the possibility of optimizing an architecture. This can take on several forms.

1. Choose the best subset of elements in **e** by maximizing A or S with respect to **e.**
2. Choose the best subset of **e** that has G or fewer elements by maximizing A or S subject to the constraint.

$$\mathbf{e}^T\mathbf{e} \leq G$$

3. Choose the best subset of **e** to maximize A subject to an upper bound budget constraint

$$\mathbf{c}^T\mathbf{e} \leq B$$

   where **c** is a vector of architecture element costs and B is the budget constraint.

4. Choose the subset of **e** that maximizes A subject to a minimum bound constraint on S.

Clearly, one could combine constraints for more complex optimizations. Given the combinatorial complexity involved, genetic optimization methods may be useful.

8.0 Application to Grid Architecture

All of the foregoing may be applied to grid architectures in particular. The structures of grid architecture are defined in terms of seven structure classes. Properties are often listed in terms of "ility" words, and can be numerous. While the electricity industry has a tendency to mix qualities and properties, the Grid Architecture discipline, these are explicitly separated. There is no wide consensus on grid characteristics (either properties or qualities) and as mentioned, "ility" words predominate in most discussions.

The following is a set of grid system qualities being developed based on the principles described above. Note that none of the qualities is an "ility". Completion of these definitions and proof of mutual orthogonality of the entire set based on their mathematical definitions are tasks yet to be completed.

1. DELIVERS – the grid provides the amount of energy customers want, when they want it, and in the form they want
2. CONSERVES – uses all resources sparingly, especially those that are not replaceable
3. PRESERVES – minimizes wastes and emissions
4. PROTECTS – provides safety or operates safely for itself, users, utility workers, and the public in general
5. ADAPTS – adjusts to changing conditions on both fast (parametric or modal) and slow (structural) time scales
6. ENABLES – provides broad access and support for customer service, energy innovation, and value realization
7. MERITS – provides sufficient useful capability and public good to support continued public and private investment

Item 1 addresses basic functionality, and so encompasses many operational excellence issues that tend to get overlooked in grid "ility" discussions. Items 2 and 3 decouple input issues from output issues, something that is often overlooked when specifying qualities like "clean and sustainable." Such decoupling leads to better definitions of unambiguous norms. Item 4 addresses the grid's responses to failures and attacks. Items 5 and 6 both address aspects of how the grid changes. Item 7 addresses economic factors but in a broader manner than affordability alone.

To give these a mathematical foundation, we need norms for each. While this work is still in progress, the following are suggestions for how this can be done for the seven qualities listed above.

DELIVERS –

$$\text{Norm} = 1/(NT)\sum_{i=1}^{N}\int_{t-T}^{t}\{\min[S_i(\tau),D_i(\tau)]/D_i(\tau)\}\,d\tau$$

S = supply
D = unmodified demand
N = number of delivery points
T = time window

CONSERVES –

$$\text{Norm} = \exp[-\underline{w}_c \bullet \underline{r}]$$

$\underline{w}_c$ = weight factors
$\underline{r}$ = vector of consumption elements
resource/MW of capacity or
resource/MW-hr

PRESERVES –

$$\text{Norm} = \exp[-\underline{w}_p \bullet \underline{g}]$$

$\underline{w}_p$ = weight factors
$\underline{g}$ = vector of waste/emission elements
waste/MW of capacity or
waste/MW-hr

thermal, spent fuel, byproducts,
noise, radiation, gases

PROTECTS –

$$\text{Norm} = \exp[-T(\lambda_f + \lambda_a)]$$

$\underline{T}$ = time window
λ_f = failure rate
λ_a = attack rate

probability of zero failures over time T

ADAPTS –

Must account for short and long time scales and for multiple simultaneous adaptations

Define adaptation over time:

s = any particular adaptation;

e = architectural element;

t = time

Then the time rate of progress of an adaptation is:

$$\partial s/\partial t = \partial s/\partial e \times \partial e/\partial t$$

where $\partial s/\partial e$ is defined to be a "flex factor." Flex factors quantify a system's ability to change with respect to a particular architectural element.

Now define the adaptation rate vector $[\partial s_i/\partial t]$ as linear transformation of the architectural element change vector $[\partial e_j/\partial t]$. For a collection of adaptations, the set of flex factors forms a matrix **F**:

$$\mathbf{F} = [\partial s_i/\partial e_j].$$

Then the ADAPTS norm is the Frobenius norm[10] of matrix **F**. Estimation of flex factors is likely to be difficult and of course would be grid and organization-specific.

ENABLES –

- α: algebraic connectivity/vertex connectivity
- β: Isoperimetric number (Cheeger constant)
- γ: open standard interfaces/total interfaces
- σ: endpoint/app scalability

$$\sigma = N/[1 + C_1 (N-1) + C_2 N(N-1)]$$

$$C_1 = \text{contention} \quad C_2 = \text{coherency}$$

$$N = \text{number of load generators or processors}$$

$$\text{ENABLES Norm} = \sum [\alpha_i] [\beta_i] [\gamma_i] [\sigma_i]$$

The approach to structural adaptation characterization is via graph theory. Algebraic connectivity is the second smallest eigenvalue of the graph Laplacian. The ratio α is intended to measure synchronizability and overall connectedness. Cheeger constant is used to measure scalability via average bottlenecking of an architectural structure represented as a graph. The interface ratio is intended to measure average ease of connection or integration of devices. Lastly, σ is intended to measure scalability for both endpoints and applications. The norm is a compounding of these individual measures.

MERITS –

This dimension is intuitive but analytically elusive. Some possibilities are Return on Equity and Social Return on Investment (SROI) but these are not rigorously defined for the purpose at hand. A proper norm must capture number of grid users, be able to incorporate time windows, and encapsulate properties like affordability and financeability (bankability).

9.0 Final Comments

In this paper we have offered a new approach to characterization of system architectures. The approach makes use of a two-stage definition of the characteristics that describe the suitability of an architecture to the purposes of both the users (qualities) and the builder/operators of the systems described in the architecture (properties). Graph theory is then used to map architecture elements to properties and properties to qualities. The graphs are translated into matrices and linear space mathematics methods are applied to develop ways to compare architectures and architectural alternatives.

Much remains to be resolved in this approach:

- Methods for computing or estimating mapping values are needed
- Proofs of the mutual orthogonality of the qualities are needed

[10] The sum of the absolute squares of the elements of the matrix, (Golub and van Loan 1996, p. 55). Also can be calculated as the square root of the trace of $\mathbf{FF}^T$.

- Definition of a norm for the MERIT quality is needed
- Tools that implement the graph and matrix representations and manipulations, as well as potentially the optimizations are needed, since real architectures are too complex for hand calculation

Completion of these items will advance the theory of system architecture by providing a rigorous approach to system architecture quantification, evaluation, and optimization.