*First*STEP: Accelerating and Perfecting Requirements

Knowledge Partners International, LLC
8 Calais Road
Mendham NJ 07945
www.kpiusa.com

Proprietary Material Notice

Table of Contents

*First*STEP: Accelerating and Perfecting Requirements

(This paper assumes knowledge of The Decision Model. If you are not already familiar with the theory of The Decision Model, you can download a brief primer from www.TheDecisionModel.com.)

Software engineering is a much younger discipline than are other branches of engineering. We see this in the continuing evolution of attitudes and approaches to requirements. In early days, there were no requirements. In later days, there were volumes of approved textual statements. Eventually, there were formal models. Today, sometimes models and statements are merely interim deliverables because code becomes the requirements.

In the spirit of this evolution, we present a requirements framework and approach that we call *First*STEP. It addresses many of the problems in business requirements today. Its three prominent characteristics are a framework, a new model, and visualization. The framework ensures completeness of *all* functional or business requirements. (We do not, in this paper prescribe the non-functional, or system aspects of requirements.) The new model is The Decision Model, transforming important business thinking into a tangible and manageable business requirement. The visualization simulates user scenarios, alleviating the need for abstract specifications or models.

Based on experience, *First*STEP accelerates the productivity and success of a software engineering project. It begins with a framework.

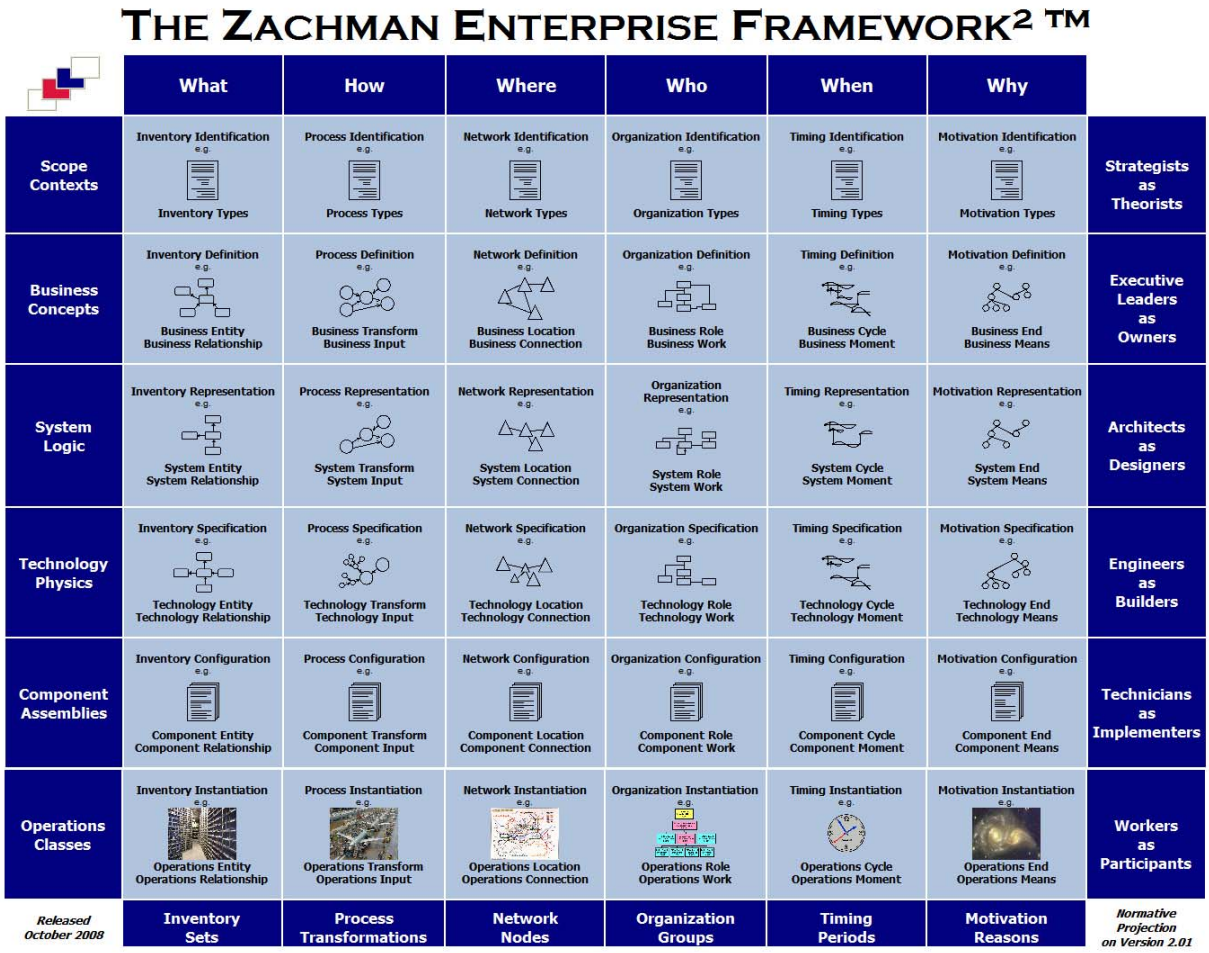
The Framework

*First*STEP adopts the Zachman Framework for Enterprise Architecture[1], which is the foundation for all modern Enterprise Architecture theory. Thus, it will likely fit into most – if not all – architectures. The Zachman Framework proposes a decomposition of complex systems, processes, or organizations. As a quick review, The Zachman Framework is a two-dimensional table, with six columns representing the dimensions which are the interrogatories (What, How, Where, Who, When, Why), and six rows representing the views of target audiences (strategists – as theorists, executive Leaders – as owners, architects – as designers, engineers – as builders, technicians – as implementers, and workers – as participants in the Functioning Enterprise). Therefore, there are 36 cells in the two-dimensional table. Each cell contains content unique to the cell, which ought not to appear in any other cell. So, it is a normalized schema of decomposition artifacts because it disallows unnecessary duplication of content among cells. Zachman calls the cell content primitives. A primitive in one cell connects to the other primitives in the same row. The Framework is set out in Figure 1.

[1] Also referred to as the Zachman Framework. Details can be found in Chapter 13 of The Decision Model (Goldberg & von Halle, Auerbach 2010)

Using this framework for requirements ensures completeness. The content of each cell is an appropriate model representing the particular dimension of the organization/system/thing represented by the framework. Each model corresponds to a particular audience view. Each model connects to other models, in the same view by common metadata and connection points.

Figure 1 The Zachman Framework



The Approach

Using the Framework, we implement requirements in five simple, but iterative steps.

Step 1: Validate the completeness of the Project Charter and Determine Scope.
Step 2: Complete the First and Skeletal Iteration of the Models (including Decision Models!)
Step 3: Visualize the target scope in Software.
Step 4: Iterate the Models (Until Complete).
Step 5: Repackage and Present a Holistic Requirements Deliverable.

Step 1: Validate the Completeness of the Project Charter and Determine Scope

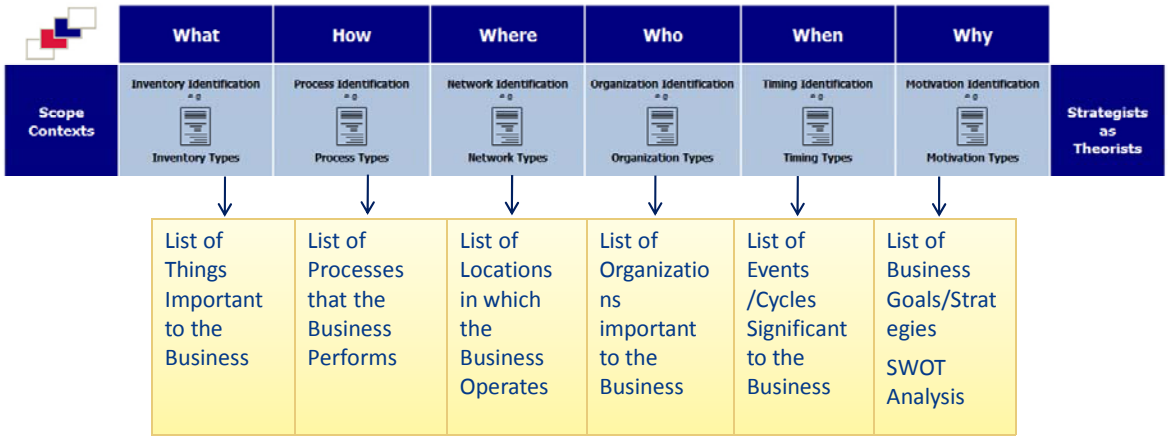
The step of validating the completeness of the Project Charter is frequently overlooked but it is the most vital first necessity, and can provide strong clues as to the scope.

For this validation, we turn to our foundation, the Zachman Framework, and begin at Row 1 in Figure 1, the perspective of the "Planner" of the system, which consists simply of six lists. Using the project charter as the scope, the lists referenced in row 1 are created:

- List of things (information)
- List of business processes
- List of locations
- List of organizations
- List of events/cycles
- List of business motivations and business decisions

If the project charter is unclear about any of these lists, request sponsors to fill in the gaps before proceeding. While the subject of scope in FirstSTEP is worthy of a separate and lengthy discussion, for the purpose of our high level review it is sufficient to say that the row 1 analysis is the basis of how we determine scope for a project.

Figure 2 Deriving Scope Using the Zachman Framework

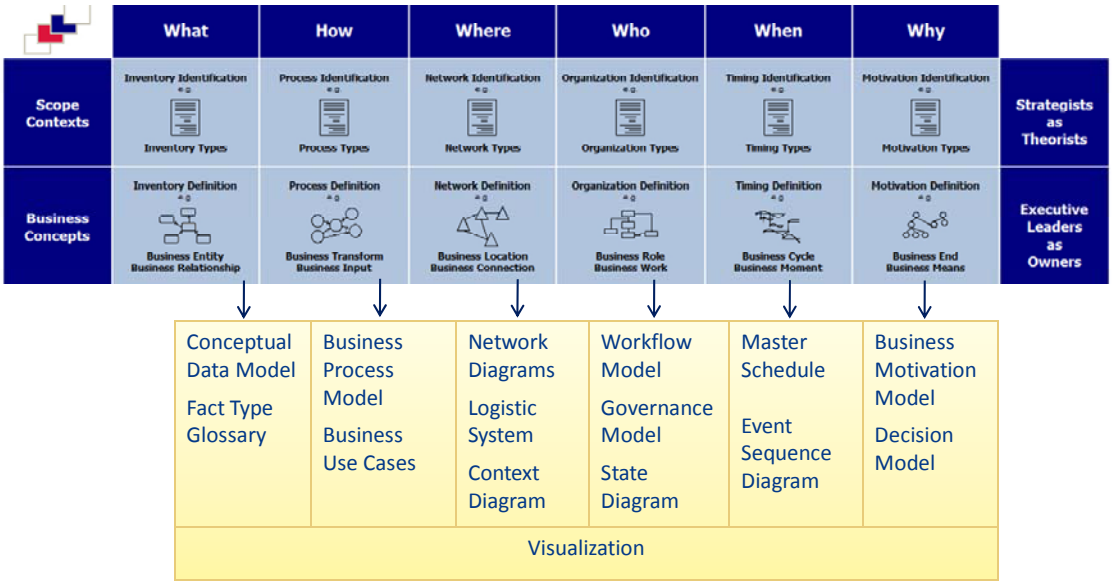


The validated scope forms the first section of the requirements document, which will be built to completion in each successive step of *First*STEP.

Step 2: Complete the First and Skeletal Iteration of the Models (don't forget Decision Models!)

The Zachman Framework is used to identify the kinds of requirements deliverables appropriate for the project. This means moving onto Row 2 because it represents the owner, that is the business audience. For each cell, the corresponding deliverable is described to the level of detail as prescribed by Zachman, that is in "excruciating detail." It is a principle of the framework that there is no difference between the detail in the cells from top to bottom, but rather a difference of perspective only. In other words it is as important for business requirements to be as detailed as the technical design is, but for different audiences.

Figure 3: Useful Deliverables for Business Requirements



Wherever possible, models are favored to represent business requirements over textual statements of functional requirements. Each model may be accompanied with descriptive text. each one aimed at a different dimension of the Framework. Figure 1 contains a list of useful deliverables for business requirements. The decision as to which of the models to employ is based upon the nature of the system being described, and is dictated by experience.

"Models provide graphical and semantic representations of their target aspects in ways that are easier to view and understand than are textual statements or narratives. Because each model captures just one aspect of the system, several models, each orthogonal to the other, provide a sufficiently complete view of the proposed system[2]. Therefore, models provide shared understanding and for assistance in developing related textual requirement statements. In some approaches, a model serves as input for the automatic generation of the system. This is called Model Driven Architecture (MDA).

There are at least three reasons for modeling each dimension with its own model. First, each model reduces complexity. Second, each model enables independent change of its dimension from other dimensions. Third, it turns out that a model specifically tailored to the characteristics of the modeled dimension best represents each dimension. For example, data is best modeled in diagrams that show the relationship of each data entity to another, where icons represent the data entities and lines represent data relationships. On the other hand, process flow is best modeled in diagrams that

[2] The operative word is "sufficiently"; after all, if we created an exhaustively complete and comprehensive model of every aspect of the system, it would no longer be a model, but the system itself! Rather like Lewis Carroll's map in *"Sylvie and Bruno Concluded",* where the mapmaker's "grandest" solution to absolute accuracy in their map of the country was to make it on a scale of a mile to the mile. Unfortunately, it could never be spread out without blotting out the sunlight, and the population had to resort to using the actual country as their map, discovering that it did nearly as well!

represent flow of process tasks, where icons represent the process tasks and lines represent process sequence." (von Halle and Goldberg, 2010)

The normalization depicted in the Zachman Framework – that is, each cell has a model that only represents the dimension of that cell – is the key to agility. It separates the dimensions of the requirements, making it possible to make changes in one dimension with minimal impact on any other. It also means that each model can most faithfully represent the nature of the underlying dimension, without the compromises that complex, multi-dimensional models have to make.

Step 2 and The Decision Model

There are several compelling reasons to include The Decision Model as a new requirements deliverable.

The first one relates to how business rules are the forgotten dimension in most requirements. Ken Orr states in the preface to The Decision Model, "Indeed, business rules are perhaps the most important, unresolved problem facing business and IT. A great many organizations are not so much the masters of their business rules but prisoners of them." That is because the business rules in those systems represent is the one dimension we have omitted from requirements, but that we now recognize as something we need to manage much better. We are unable to do so because we have not had a separate deliverable for them.

The good news is that The Decision Model fills this gap as a stand-alone model just for organizing business rules similar to how the Relational Model is for data. In The Decision Model, business rules are in their atomic form, governed by 16 principles, including three forms of normalization. The inherent structure of business logic in The Decision Model is similar to the inherent structure that Dr. Codd detected in data almost 40 years ago. This is a very exciting development for requirements practitioners.

A second, even more compelling reason to include The Decision Model is its potential impact on the business itself. Practice proves that when business leaders can view their business logic in a tangible, rigorous Decision Model structure, it becomes a magnet for better business decision-making. This has a profound effect not only on requirements, but also on organizational maturity and business performance.

And finally, the third reason is that The Decision Model shares metadata with, and has connection points to, the other models commonly used in requirements today. This provides the natural connection between the models on row 2, and greatly enhances the value of the other models. Some examples of this are, The Decision Model provides:

- A means to separate business logic out of process models, greatly simplifying those models, enabling the business to better described the processes and create more maintainable – and understandable – process models.
- Separation of the business logic from the business process allows a greater traceability, control and management of business motivations by the business, resulting in Business Decision

Management (BDM), where the business has greater flexibility of the operational decisions in the management of the business.

- A natural connection between the business glossary (and the conceptual data model) and the business logic through its rigorous use of the glossary, and by its clear definition of derived data. Most requirements do not consider derived data, which in many cases is the most important data used in the business, because without The Decision Model, such data is not to be discovered.
- Significant opportunities to separate the governance and maintenance of business logic from the process, allowing a greater granularity of business and audit control.

Step 2 and Model Iteration

Step two includes a first and skeletal iteration of the models. These are at a high level so that the model selection can be validated as appropriate. The level of detail for this first iteration is determined on a model by model basis, but generally only sufficient level of detail to understand the appropriateness of the models across the row of the Zachman Framework as the basis for complete requirements.

For example, process models are developed to a level of detail that exposes business decisions. T then the business decisions are modeled in A Decision Model in its "skeletal" form, so that the high level logic of the business decisions is understood and the principle derived data is discovered. This allows for the beginning of the business glossary and a high level conceptual data model. Also, the business motivations for each business decision are identified, and other models such as the state models, data flow, time sequence, logistic networks, and governance may also be completed in skeletal form. This iteration is complete when the models most appropriate to the requirements are completed to an initial level of detail.

Often, this first iteration leads to an understanding of more appropriate models for a given cell in the Framework. Importantly, it also provides the means to re-assess scope in greater detail, and with more accuracy, once the high level models have been completed.

During the building of the first iteration of models, requirements identified but not able to be represented in the models are captured and entered into the requirements document. More on how this is accomplished is discussed in Step 5.

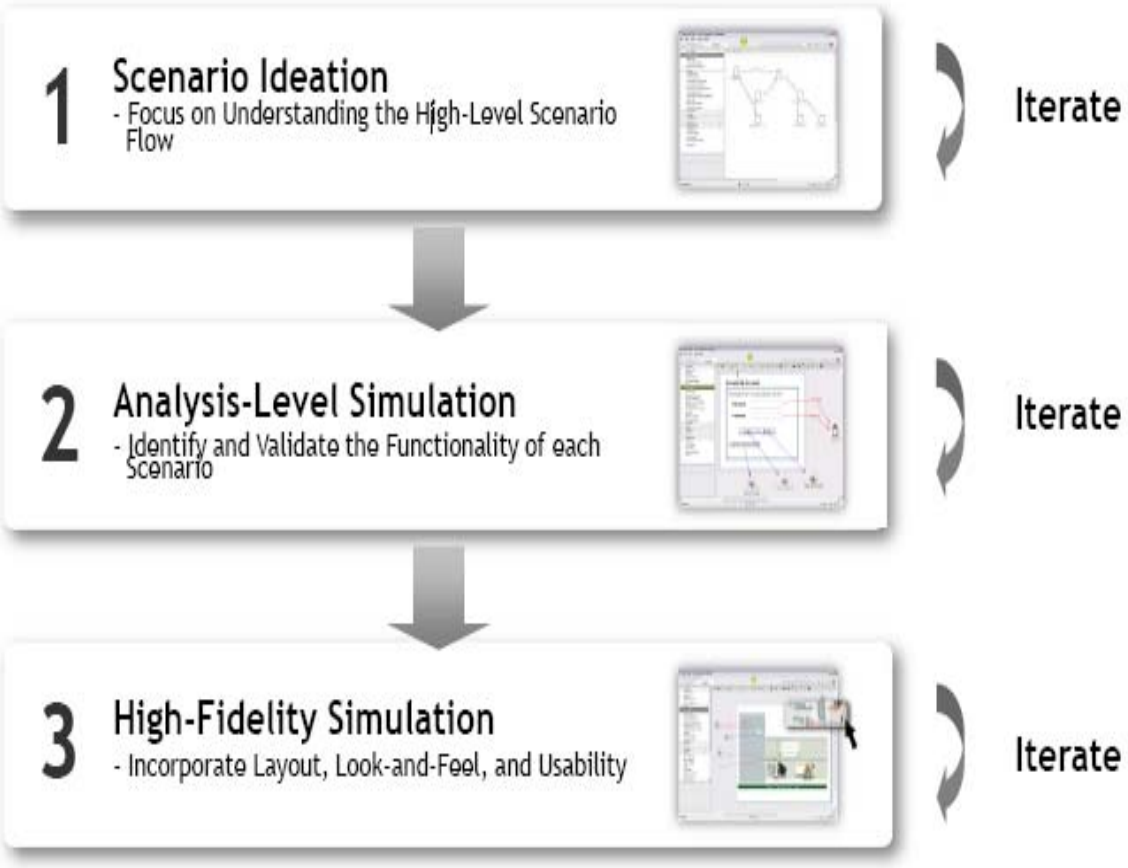
Step 3: Visualize the Target Scope in Software

Visualization is the simulation of important user scenarios of the system. It enables exhaustive examination of business requirements. The visualization should take place after the first iteration (Step 2), and before finishing models because it reveals requirements arising from these interactions. Visualization takes place in three distinct phases, as illustrated in Figure 4. A software product that we have used with great effect in creating the visualization is Irise, which is designed specifically to create high fidelity, interactive visualizations.

The first phase of Visualization is Scenario Ideation. It divides the requirements into a set of scenarios generally grouped around a set of logical functions. It explore each set of scenarios through steps until

all scenarios are complete. In some cases, users may decide that there is no value to creating a high-fidelity simulation, and that an analysis level simulation is sufficient.

Figure 4: High Level Phases of Visualization



During the Scenario Ideation and the second phase, Analysis-Level Simulation, participants identify a large number of requirements. A wider group of participants discovers more when given access to the simulations. This leads to more exhaustive discovery of requirements that may not have been identified using conventional methods. It also leads to higher levels of adoption of the product.

Textual requirements are captured in the visualization tool, and traced – where appropriate – to features in the User Interface. Where necessary, traceability to the appropriate models is also noted. These textual requirements will be generated into a document that integrates into the master requirements document, while retaining the traceability.

The third phase, High-Fidelity Simulation, delivers a refined, realistic prototype. Stakeholders explore it for effectiveness and use it in considering the completeness and acceptability of the requirements in the visualization. Often, this exposes additions or changes to the outline models already created.

Such changes to the models may be as simple as data attributes not formerly identified, or as complex as a significant number of process steps, or business decisions missed in the first iteration. This discovery

and rising level of confidence leads to greater buy-in by all the stakeholders, and still higher levels of adoption.

The Visualization step, like all the steps in *First*STEP is expected to be done iteratively: that is, a particular phase may be repeated when additional levels of detail are detected.

Step 4: Iterate the Models (Until Complete)

George Box, industrial statistician famously said "Essentially, all models are wrong, but some are useful" (Box 1979). This points to the fact that even the most rigorous, most complete, and most accurate set of requirements may – at best – be merely useful, and not "right". Models are but an aspect of the system, not the system itself. But we do not dismiss models simply because they cannot be 100% accurate – we make them as useful as possible. (Because we shouldn't, like Lewis Carroll's mapmaker, blot out the sun with our map, means our less massive map may be more useful than the "100% accurate" one). Making a model useful requires rigor in our model making – Zachman calls it "excruciating detail" – achieved through an iterative process, where detail in one model may lead to greater detail in another, and so on in a series of successive steps.

For example, adding logic to the skeletal decision models often leads to the need for sequence between logical steps. For example a Rule Family may result in multiple conclusion values, requiring a process task to iterate through those multiple values to determine the appropriate result. Consequently process models which use that particular Decision Model need adjusting, and the necessary process tasks added. Conversely it may be discovered that several activities in a process model are not really sequential tasks at all, but represent declarative logic, and are best represented in a Decision Model. Changes in the process and decision models may lead to consequential changes in the glossary and in the data flow model, and so on.

Only by the completion of all the models in the row is it possible to establish completion of any of the individual models. The inter-related nature of the models ensures that completeness of the whole can be determined by testing the completeness of each. The iterations end when there are no further outstanding issues on any of the models.

During this step, as in all the previous steps, narrative is developed to give additional context to the models. This narrative forms part of the requirements document, in which the models are imbedded. Requirements identified, but not sufficiently well captured in the model are developed into textual requirements statements, which are also maintained in the requirements document. Step 5 below deals with these textual requirements statements in greater detail.

The models may also serve as the basis for test case development. This can be done as a part of the requirements phase, or in a subsequent phase, depending on the methodology being adopted. Models particularly lend themselves to scenario testing methods, and The Decision Model provides an opportunity for a testing approach that allows the actual logic in the model to be tested directly.

Step 5: Finalize the Requirements Deliverable.

Consistent with the framework, the requirements document is organized into six sections – one for each column of the Zachman Framework. Each section of the document contains the appropriate model, together with the accompanying narrative and textual functional requirements statements.

Little has been said thus far about the textual requirements statements. These are clearly important components of the requirements, being necessary to bring into sharp focus that which may not be sufficiently clear in the models. It is, of course, highly preferable that the models are comprehensive to the point of limiting the need for such textual statements for functional requirements. However, we recognize that there will always be the need for these statements. It is important that the same level of rigor is applied to the textual statements that are applied to the models. This allows traceability, maintainability and agility in the requirements for the future. *First*STEP provides guides and standards for the formulation of textual requirements statements, as well as templates for the Requirements deliverables.

Summary

There is a need to improve the quality of requirements gathered in projects developing software systems for business because a large proportion of projects fall short or fail due to inaccurate or missing requirements.

The requirements approach in *First*STEP combines common practices with visualization and The Decision Model. The most important parts of this *First*STEP are:

- Select a requirements framework that represents complex systems or organizations as discrete dimensions.
- Use the framework to identify the requirements deliverables for your business audience.
- Deliver models for each dimension of your project.
- Include The Decision Model for important business thinking behind business decisions.
- Adopt visualization to elicit requirements before completing models.
- Organize the requirements document according to the framework, packaged with the High-Fidelity Visualization.

*First*STEP has also been used very effectively in Agile methodologies. In Agile, “Big Requirements Up Front” (BRUF) is discouraged, and formal documentation is eschewed. But *First*STEP provides techniques to support and strengthen an Agile approach. In Agile, the first four steps are adapted to be even more iterative, where the modeling is more akin to the “model storming” approach of Agile, and the fifth step is replaced by the coding and testing phase of development. Agile proponents may disagree with the use of visualization, insisting that this step is done in the coding phase. However, we have found that the adaptation of visualization into the Agile process significantly reduces risk and in fact may contribute to a more efficient process than using the classical Agile approach.

The *First*STEP approach is independent of the methodology used in the system development lifecycle itself. It may be used in either a waterfall or an iterative system development lifecycle. It is also

independent of technology used to store or manage the requirements, or of elicitation methods. However, requirements gathering technology can add significantly to *First*STEP, particularly in the modeling and reuse of artifacts, and in the traceability across artifacts.

*First*STEP provides, in addition to ensuring completeness of requirements, an improvement of the traceability of requirements to the design and build artifacts used in the development and maintenance lifecycle of the system. This is particularly true when the architecture of the final system maintains a separation of concerns in line with the Zachman Framework. This, after all, is consistent with good architectural practice, and will lead to more maintainable, more agile solutions.