

## Using the Zachman Framework to Assess the Rational Unified Process

by [DJ de Villiers](#)

Technical Representative  
Rational Software

*Since its publication in 1987, the Zachman Framework has provided organizations with an effective way of assessing the completeness of software development process models, in terms of an organization's information needs. This article provides a Zachman-Framework-based analysis of the Rational Unified Process (RUP), a software development process model that describes a use-case-driven, architecture-centric, and iterative approach to developing quality software -- on time and within budget. Organizations considering implementing the RUP can use this analysis to better understand how the (many) deliverables defined by the RUP relate to their information needs.*



### Overview of the Zachman Framework

In his 1987 paper, "A Framework for Information Systems Architecture," John Zachman refers to the increasing difficulty of managing information systems. Some of the contributing factors he identifies are the increasing size and complexity of these systems, coupled with the tendency to distribute them as more enterprise operations are automated. He then emphasizes the importance of information systems architecture in bringing structure to these decentralized systems.

According to Zachman, good information systems architecture is derived from the information system strategy, which, in turn, is determined by the business strategy. He implies, therefore, that the information systems should support the business objectives of the enterprise, but he does not expand on a strategic planning process. His model describes the deliverables (*artifacts*) of a software engineering process. He identifies two different kinds of representations which, when used in combination, precisely describe the nature and purpose of these deliverables within the

- ▶ [subscribe](#)
- ▶ [contact us](#)
- ▶ [submit an article](#)
- ▶ [rational.com](#)
- ▶ [issue contents](#)
- ▶ [archives](#)
- ▶ [mission statement](#)
- ▶ [editorial staff](#)

organizational context.

### **First Representation: The Human Perspective**

The first representation addresses the different perspectives, or views, of the people involved in information system development. Zachman derives these perspectives by analogy from the models used in the construction and product engineering industries. Thus, he identifies three fundamental architectural views necessary for the development of a product:

- Owner's view;
- Designer's view;
- Builder's view.

He then identifies three more views to accurately depict all development stages for the information system artifacts within the enterprise:

- Scope, or ballpark, view;
- Out-of-context view;
- Operational view.

Each view is a description from a particular perspective and has a representation (a model or functioning system), as indicated in Table 1. Here is a brief description of each view and model/functioning system:

- **Scope (Ballpark) View.** This view describes the business purpose and strategy, which defines the playing field (ballpark) for the other views. It serves as the context within which the other views will be derived and managed.
- **Owner's View (Enterprise Model).** This is a description of the organization within which the information system must function. Analyzing this view reveals which parts of the enterprise can be automated.
- **Designer's View (System Model).** This view outlines how the system will satisfy the organization's information needs. The representation is free from solution-specific aspects or production-specific constraints.
- **Builder's View (Technology Model).** This is a representation of how the system will be implemented. It makes specific solutions and technologies apparent and addresses production constraints.
- **Out-of-Context View (Detailed Models).** These representations illustrate the implementation-specific details of certain system elements: parts that need further clarification before production can begin. This view is less architecturally significant than the others because it is more concerned with a part of the system than with the whole.
- **Operational View (Functioning System).** This is a view of the

functioning system in its operational environment.

A common misconception is that there is a progression within the two sets of views from abstract to more detailed representations of the product. Zachman emphasizes that each view is distinct and has a unique nature and purpose. Within each of these views the level of detail can be adjusted accordingly. What is useful to the owner might not be at all useful to the builder, no matter how detailed it is. Similarly, the owner might not be able to understand the designer's models, irrespective of the level of abstraction, simply because they describe a completely different view of the system.

### **Second Representation: Six Foci**

The second product representation distinguishes different foci of the system. These foci are used to classify relevant characteristics and emphasize different aspects of the system. Each focus provides a different, product-centric view of the nature of the system, which identifies related characteristics and suppresses unrelated ones. Although these different views all relate to the same system, they remain isolated from each other:

- **What.** This describes the system contents, or data, in the case of information systems.
- **How.** The usage and functioning of the system are described here, including processes and flows of control.
- **Where.** Spatial elements and their relationships are represented in this view, including system distribution.
- **Who.** This depicts the people and organizational units interacting with the information system.
- **When.** This view brings sequencing and timing to the processes and flows described in the *How* view. These two views are closely related to each other yet remain distinct to avoid unnecessary complexity. Concurrency is also addressed in this view.
- **Why.** Motivations for creating the system and rules for constraining it are highlighted in this view. These constraining rules will most often be applied to the *Why* and *How* descriptions.

### **Combining the Two Dimensions**

By combining these two dimensions of representation into a matrix (Table 1) Zachman created his Framework, a powerful tool for analyzing software engineering deliverables. The distinct nature of the different representations along each of the axes makes it possible to precisely describe the nature and purpose of each cell. Using this description, an organization can assess the coverage of its software development process and evaluate the impact of new tools and techniques within the context of both business and information system strategies.

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
<b>Scope View</b>	List of things important to the enterprise	List of processes the enterprise performs	List of locations where the enterprise operates	List of organizational units	List of business events/cycles	List of business objectives
<b>Owner's View</b>	Entity relationship diagram	Business process model (physical data flow diagram)	Logistics network (nodes and links)	Organizational chart, with roles, skill sets, security issues.	Business master schedule	Business rules
<b>Designer's View</b>	Data model (converged entities, fully normalized)	Essential data flow diagram, application architecture	Distributed system architecture	Human interface architecture (roles, data, access)	Dependency diagram, entity life history (process structure)	Business rule model
<b>Builder's View</b>	Data architecture (tables and columns), map to legacy data	System design: structure chart, pseudo-code	System architecture (hardware, software types)	User interface (how the system will behave), security design	"Control flow" diagram (control structure)	Business rule design
<b>Detailed View</b>	Data design (denormalized), physical storage design	Detailed program design	Network architecture	Screens, security architecture (who can see what?)	Timing definitions	Rule specification in program logic
<b>Operational View</b>	Converted data	Executable programs	Communications facilities	Trained people	Business events	Enforced rules

**Table 1: The Zachman Framework**  
(View [full size graphic](#) in new window)

## Overview of the Rational Unified Process

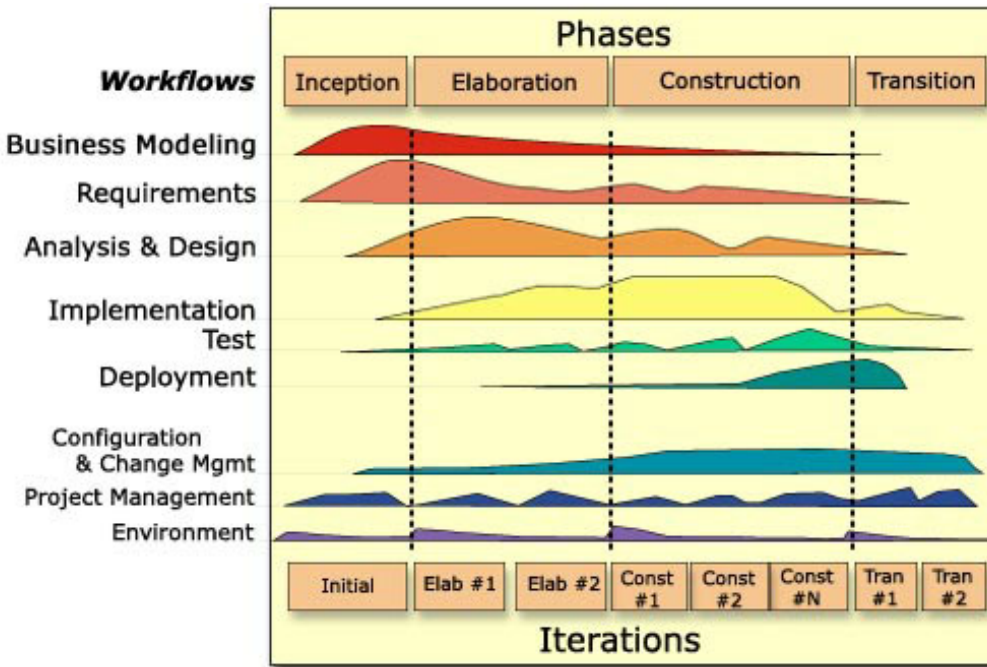
The Rational Unified Process (RUP) is a use-case-driven, architecture-centric, iterative, and incremental software development process. Its goal is to enable software development teams to produce high-quality software that meets user needs -- on time and within budget.

The Rational Unified Process captures a number of modern software engineering best practices in a tangible and practical form. These best practices have been identified by the industry as major contributors to the success of many software projects. They are:

- Develop iteratively
- Manage requirements
- Model visually
- Use component-based architectures
- Continuously verify quality

- Control changes

The Iteration Model Graph shown in Figure 1 provides an overview of the process, emphasizing its two dimensions: time along the horizontal axis and structure along the vertical axis. This two-dimensionality distinguishes the Rational Unified Process from many other process models, which use a single dimension to illustrate development lifecycle phases.



**Figure 1: Rational Unified Process Iteration Model Graph**

The Rational Unified Process's structural dimension describes the software development lifecycle in terms of *roles*, *artifacts*, *activities*, and *workflows*.

- Roles describe a cohesive set of responsibilities that a person may play in the development process. Examples of roles are Architect and Design Review Board.
- An artifact is a deliverable that is used, produced, or modified during the development lifecycle. Examples of artifacts are Use cases, Design model, and Risk list.
- An activity is a unit of work, performed by one or more persons playing a role, which produces or modifies one or more artifacts. Examples of activities include Determine system boundaries and Fix a defect.
- A workflow is a collection of functionally related activities and the roles and artifacts related to these activities. Within a workflow, we can see which roles perform which activities with which artifacts. The workflow also outlines the sequence of these activities, so that we know when they must occur. Examples of workflows are

## Phases and Iterations

The dynamic dimension of the Rational Unified Process describes the development lifecycle in temporal terms, using phases and iterations. The four phases are *Inception*, *Elaboration*, *Construction*, and *Transition*.

- *Inception* is the phase in which the scope of the project is determined and the business case is defined.
- *Elaboration* focuses on defining and validating a robust architecture to eliminate technical risk and uncertainty.
- *Construction* is the phase in which the product is built. Functionality is incrementally added to the architectural baseline.
- *Transition* is the phase in which the product is delivered to end users.

Each phase ends with a project milestone; this includes a review and a decision as to whether the project can safely proceed to the next phase. A phase, therefore, is not a distinct step during development, as is analysis or implementation. Rather, a phase describes 1) what the focus of activities should be at a specific point during the product lifecycle, and 2) the evaluation criteria for determining whether the phase has been successfully accomplished or not.

Phases consist of one or more iterations. The number of iterations is determined by how rapidly and accurately the milestone for that phase can be reached. For example, if the problem is vague and the development team has little domain experience, more iterations will be necessary during Inception. This will ensure that the team can reach an accurate description of the problem and that they will not encounter domain-related surprises later. A project that involves a new, unproven technology will have more iterations during Elaboration, to ensure that the architecture incorporates the technology in a sufficient and stable manner.

Every iteration is a miniproject in its own right because it includes activities from all the core workflows: Business Modeling; Requirements; Analysis and Design; Implementation; Test; and Deployment. The phase the iteration belongs to dictates how much emphasis needs to be placed on each of these workflows. Having said this, it might be useful to refer back to Figure 1. We can see that the Inception iterations will include activities from all workflows, but will focus more on business modeling and requirements activities. Construction iterations, on the other hand, will focus much more on design, implementation, and testing, although some requirements activities may be included.

## An Artifact-Driven, Risk-Mitigating Process

One of the Rational Unified Process's important characteristics is that it is artifact driven. To this end, the core workflows are centered on the creation and validation of models, and each model is a complete























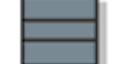


representation of the system from a different viewpoint. Models are artifacts, and therefore they are also software engineering deliverables. A benefit of a model-driven approach is that the information produced during development is captured and maintained in its most useable form, eliminating the need to constantly reformat and rework information.

Another feature of the Rational Unified Process is that it is a risk-mitigating process. Every iteration involves risk analysis and prioritization, and is planned around the mitigation of the most significant project risks. This is one of the principles of iterative development. In most cases the business risks will be assigned a higher priority than technical risks, but this can vary, depending on project circumstances.

## **Analyzing the RUP as a Process Description**

If we consider the Rational Unified Process as a software development process, we can apply the Zachman Framework to determine its effectiveness in terms of its coverage of software development deliverables. We should first reorder the columns of the Zachman Framework, however, because the RUP has both static and dynamic dimensions. This minor adjustment has no effect on the Framework itself because there are no dependencies between columns, but it does make the results more readable.

Because the Rational Unified Process is artifact driven, we want to map the defined artifacts onto the Zachman Framework, but in fact there is no way to do precise mapping. As Table 2 shows, some artifacts address more than one cell in the matrix -- sometimes disjoint cells -- and some combinations or sets of artifacts address only one cell. For more information regarding the meaning, purpose, and contents of the artifacts listed in Table 2, refer to the Rational Unified Process. Note that some cells clearly identify specific parts of artifacts to which they refer.

	Motivation (Why)	People (Who)	Function (How)	Data (What)	Time (When)	Network (Where)
Scope View	 Vision:Needs	 Vision: Stakeholders	 Vision: Features	 Business entities	 Business workflows	
Owner's View	 Business rules	 Actors	 Use cases	 Business object model	 Use case flow of events	 Network configurations
Designer's View	Constraints, multiplicities, workflow activities	 Boundary classes	 Use case realizations	 Persistent classes	 Interaction diagrams	 Deployment model
Builder's View		 End user support material	 Components	 Data model	 Process model	 Process-to- node mapping
Detailed View		 UI design classes	 Design classes	 Columns, types, keys, indexes	 State machines	

**Table 2: Analysis of the Rational Unified Process as a Process Description**  
(View [full size graphic](#) in new window)

Table 2 does not include the Zachman Framework's standard bottom row (*Operational View*) because this row is process independent. The functioning system is the product under development, so its elements (business events, for example) will be present no matter what process model you use, and regardless of whether the process model explicitly refers to them.

To map a dynamic dimension onto the results presented in Table 2, we would somehow have to determine a path through the matrix that defines the sequence in which the artifacts are to be produced (or refined). Common sense tells us that the path would have to travel downwards because we couldn't possibly produce a physical view of something without first making a conceptual representation. The relationships among the artifacts, roles, and activities in the Rational Unified Process imply relationships between the matrix's columns, and that is why we reordered the columns earlier. Roughly speaking, the path would also have to make its way from left to right, as we successively explore the motivations, the people involved, the behavior, the structure, and so on.

The exact path might travel downwards one row at a time while moving from left to right one column at a time, or it could travel left to right one column at a time while moving from top to bottom, row by row. These two approaches relate loosely to the "wide and shallow" versus "narrow and



deep" iteration strategies described in the Rational Unified Process. Furthermore, bearing in mind the iterative nature of the process, it is quite feasible that the exact path could differ across iterations. In effect, this would add a third dimension to the matrix, with each plane representing a specific iteration. Obviously, the Zachman Framework is not really designed to represent a dynamic dimension, so we would need to decide on our own the best way to present this information.

## **A Meta-Analysis of the RUP as a Process Framework**

Since the Rational Unified Process is also a software development process framework, we can use the Zachman Framework to analyze it at the meta-level. Because the product we are analyzing is the process *itself* rather than an information system, we will use the following representations to describe the RUP in terms of the Zachman Framework:

- *What* indicates the deliverables (or artifacts) of the development process.
- *How* indicates development activities.
- *Where* depicts locations at which development occurs.
- *Who* describes the roles involved in developing the software.
- *When* indicates the chronological ordering of development activities.
- *Why* provides motivations and constraints relating to the software development process.




























Table 3 shows the results of mapping the Rational Unified Process meta-artifacts onto the Zachman Framework. We see that the *Where* column is not addressed at all because the spatial distribution of software development cannot be specified at the meta-level. At the most, guidelines and suggestions can be provided for managing distributed development, but these remain independent of any context.

Another challenge is the motivation (*Why*) column. Each organization has its own specific reasons for wanting a software development lifecycle model, although the objectives are similar across organizations: to develop software that meets user needs, on time and within budget. Rational has summed up these common objectives at the highest level with the "e-software paradox": the need to balance conflicting demands for quality (user needs) and speed (time and budget). At the conceptual level, these objectives can be codified as best practices -- proven techniques that every organization should adopt as goals. Below this level, however, the rules or constraints of software development cannot be specified without context.

The Rational Unified Process describes workflows in terms of roles, artifacts, and activities. Due to the artifact-driven nature of the RUP, the selection of artifacts for a particular instance of the process, or development project, drives the selection of roles and activities. This implies relationships between the *What*, *Who*, and *How* columns that

cannot be represented within the Zachman Framework.

The bottom row (*Functioning System*) of a meta-analysis represents an instance of the software development process (a development case). Artifacts listed in this row are not meta-level concepts but rather tangibles that are used to manage different aspects of the software development process. This indicates an important distinction between the Rational Unified Process and many other lifecycle models: It enables you to manage not only the project, but also the process itself. A process model that does not include this meta-level management capability will not have any artifacts to map into this last row.

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Scope View	 Artifact overview	 Activity overview		 Role sets	 Phases & iterations	 Speed & quality
Owner's View	 Artifact	 Activity		 Role	 Core workflow	 Best practices
Designer's View	 Artifact details	 Activity steps		 Role details	 Workflow details	 Concepts
Builder's View	 Artifact guidelines	 Work guidelines		 Role responsibilities	 Activity steps	
Detailed View	 Checkpoints	 Tool mentor		 Staffing guidelines		
Functioning System	 Development case: Artifact classification	 Development case: Workflows & activities		 Staffing plan	 Iteration plan	 Vision: Problem statement

**Table 3: Meta-Analysis of the Rational Unified Process as a Process Framework**  
(View [full size graphic](#) in new window)

## Limitations of the Zachman Framework

If you were to map the Zachman Framework *itself* into the Zachman Framework (an activity beyond the scope of this article) and then compare the resulting matrix with the Rational Unified Process matrices, you would see a striking similarity between the two models. Artifacts play a central

role in both models. Although Zachman's model does not describe columnar relationships, it does describe information systems architecture in terms of deliverables. The RUP also does this and more, providing detailed descriptions of the relationships among artifacts and roles and activities. The Zachman Framework is most useful for this form of analysis: It provides a high-level summary of relevant development artifacts. In this respect it shows the RUP to be satisfactorily complete, both as a process description and as a process framework for development.

**It is important to keep in mind, however, that the Zachman Framework cannot really assess the full capabilities of a software lifecycle model such as the RUP, which describes software development from both static and dynamic points of view. The static view addresses artifacts and their relationships to one another, plus roles and activities and their relationship to artifacts; the dynamic view describes how all these static aspects relate to each other across the life of the project. Because the Zachman Framework addresses only the first column (*What*) of software development, it can present only a static view of software development deliverables that neither shows nor implies relationships among them. So it cannot analyze the relationships among phases, activities, roles, and artifacts across the lifecycle. Nor can it analyze the planning and review cycle or capture the iterative and risk-sensitive nature of the RUP.**

If you were to compare the RUP with another lifecycle model XYZ using the Zachman Framework, for example, the two processes might seem very similar if they defined similar deliverables (artifacts). In truth, however, the XYZ process might be activity driven, whereas the RUP is artifact driven. This subtle yet important difference might make the overall effectiveness of these two models radically different, but this would not be evident from the analysis. Organizations should be aware of this limitation when using the Zachman Framework as an analysis tool.

---

*My thanks to Philippe Kruchten, Kurt Bittner, Terry Quatrani, Catherine Southwood, Marlene Ellin, and Mike Perrow for their input and help with this article.*



**For more information on the products or services discussed in this article, please click [here](#) and follow the instructions provided.**

**Thank you!**