from The Rational Edge: This article proposes innovative ways to combine three of the most important methodologies that have emerged in the past decade in the field of information systems architecture: UML, RUP, and the Zachman Framework.

Over the past decade, the advantages of using the Unified Modeling Language (UML) for modeling software applications have become clear. During this same timeframe, the Rational Unified Process, or RUP, has proved itself as a software development process, and the Zachman Framework\(^1\) has proved itself as a framework for organizing and communicating architectural artifacts. Amid the swarm of overlapping methodologies, these stand apart as three pillars of modern information systems architecture. This article considers the combined usages of these technologies by examining their meta-characteristics and proposing some ways to apply them in combination within organizations.

**UML, RUP, and Zachman in brief**

By definition, UML is a modeling language. It attempts to standardize graphical language elements for modeling software-intensive systems. Its latest version, UML 2.0, consists of thirteen diagram types used for structural, behavioral, and interaction modeling. It is worth noting that, although primarily intended for object-oriented (OO) analysis and design, UML can be used under a variety of other conditions. For example, the UML Use-Case methodology (illustrated in Figure 1) is not an OO process per se, but is arguably the best technique for general functional requirements analysis. Other UML diagrams, such as Interaction, State, and Activity, are likewise handy, universal tools for describing just about any real-world project situation.
Figure 1: A UML Use-Case diagram

About Zachman

An architecture framework is a tool that can be used for developing and maintaining a broad range of architectures. When it comes to supporting a customized enterprise architecture (EA) function in the organization, the Zachman Framework is a great aid. It classifies enterprise subjects into a 6x6 matrix of cells, with each cell representing a unique view of an organization (see Figure 2).
Columns within Zachman represent the most important enterprise aspects (data, function, network, people, time, motivation), while rows differ by the perspective (scope, business, system, technology, details, assets) and group (planner, owner, designer, builder, subcontractor) of interest relative to an aspect. Incrementally, rows also differ by the level of detail as they represent abstractions (contextual, conceptual, logical, physical, detailed, and actual) of the enterprise, which may, in turn, be linked with groups and perspectives to form a grid of enterprise models and responsibilities. Example is: "A system model (perspective) is a logical (abstraction) entity that is a part of a designer (group) scope of responsibilities."

The thirty-six framework cells can be populated with models and artifacts describing the enterprise according to select meta-characteristics, such as an aspect, a detail level, or an interest category. The framework does not dictate or prescribe a notation or an order in which cells are to be filled, as this knowledge lies beyond its goal of providing a reference structure. (This latter assumption supports making the case for using the framework in harness with UML and RUP.)

**About RUP**

A process can be defined as "a series of actions, changes, or functions bringing about a result." RUP is a process framework that allows project workflows and tasks to be organized into series of activities, with the end goal of delivering a software product or a solution (see Figure 3). RUP should be tailored to create organization- or project-specific process instances and methodologies that meet the organization's specific needs.
The idea of RUP arose from the realization that a unified system for delivering artifacts using UML notation was needed. It was essential that the new process be iterative, architecture-centric, and requirements-driven, features that existing systems were short on.

Since its emergence, RUP has evolved from a software engineering process rooted in the Objectory method into a flexible and fully customizable platform for process tailoring based on UML 2.0 and supported by the Rational Method Composer (RMC). For example, using RMC, a skilled process engineer can create an instance of a systems engineering process, customize its structure, add required content from other industry-standard methodologies and best practices, and generate a ready-to-use, organization- or project-tailored instance of a process in the form of hyperlinked documentation.

Drivers for integration

Each of the three systems under discussion, having been created to serve specific needs, embody shortcomings that constrain wider applicability outside their realms of focus. One thing I noticed about the Zachman Framework is that, in many organizations, it literally acts as a great poster that everyone admires, but no one uses. I attribute this to the fact that, as with any other static framework, Zachman does not address how artifact intake should be handled. Another practical weakness, which is often promoted as an advantage, is the framework’s lack of a standard notation.

The problem with applying RUP more generally lies with its main strength of being a software development process. And while this does appeal to architects and developers alike, people from other disciplines simply do not buy it as their core methodology. For example, project managers prefer other methods (like The Project Management Book of Knowledge (PMBOK), which the RUP
Project Management discipline is based on) and treat RUP as an adjunct to these more favored methodologies. Enterprise architects, in turn, often prefer Zachman.

Role dependence is somewhat less of an issue with UML because, unlike RUP, it does not suggest roles, thus removing a need for methodology mapping. However, this process and framework independence also limits UML's usefulness. Fortunately for UML, it enjoys nearly universal acceptance, despite the fact that many organizations use other structured methods and notations like the Entity Relationship Diagrams (ERDs) and the Business Process Modeling Notation (BPMN).

In attempting to address some of these limitations and find common applications for the three methodologies, it is relevant to note once again that they have been created to address different problems within the same domain (information systems architecture), and the objectives set in front of them imply virtually no functional overlap (see Figure 4). This is actually good news for both enterprise and project architects, as it implies that UML, Zachman, and RUP can be used jointly to deliver better overall enterprise value.

![Figure 4: Functional overlap between UML, RUP, and the Zachman Framework](image-url)

**Thoughts on cross-function applicability**

RUP already makes use of UML, as well as other visual modeling languages such as ERDs, and non-visual modeling work products such as use cases, business rules, and acceptance tests (to name a few). The Zachman Framework incorporates neither a notation nor a process, nor does it impose limitations on their selection. Situations where the two technologies can best be used are also different. For example, it would be questionable to use Zachman in a development environment by virtue of its inherent unsuitability. Likewise, using RUP as the foundation for an enterprise information architecture would be equally unjustified.

The similarities observed between their respective lifecycle definitions (evident when comparing Figure 2 and Figure 3 above) lend themselves to the fact that both RUP and Zachman are artifact-driven and inherit their structure from principles of architecture-centric design. The noted similarities imply that the two processes may be mutually enriching given the ever-growing complexity of enterprise systems and projects.

Having thus far devoted considerable discourse to pointing out the differences of the three technologies, I'll now look at where the methodologies in question can potentially complement one another.
Application 1: Consider UML for the role of common enterprise notation

Usually, an organization directs its attention toward Zachman in response to a management "call for action," to drive a "to-be" model for enterprise IT systems, or to address a looming business transformation project. In scenarios such as these, architects pull together pieces of system knowledge scattered across corporate repositories (which is, essentially, the right thing to do) only to learn that artifacts produced by their predecessors do not fit well into the Zachman structure. A common reason for the mismatch is that most existing artifacts are hybrids of at least several different aspects (Remember the Zachman columns?). The other reason is that the artifacts are often created using different notations. While in most cases, it is not feasible to retrofit the artifacts, the "lesson learned" relates to the importance of a common notation. Are you thinking what I'm thinking? That's right -- UML may have a role to play in resolving this difficulty.

Zachman neither dictates nor recommends UML or any other notation. RUP, however, does require the use of UML. Since the use of multiple notations should be avoided, the latter assertion would seem to strengthen the case for leveraging UML as the common notation for both enterprise and project architecture.

Application 2: Use UML to bond Zachman and RUP

In most organizations, the lack of a standard notation is quite understandable since, while its systems might have existed in some form for a very long time, UML, for example, has been around for a mere decade. Even in the context of relatively modern systems, UML might not have been used for reasons ranging from time limitations to document designs to a shortage of the necessary skill sets.

Apart from a general need for a good modeling language for analysis and design, either of two factors may trigger the need for UML in the organization: a RUP project or an enterprise architecture modernization effort. Either reason is good as it allows an organization to kick-start UML. However, the results produced during RUP projects are often superior from the standpoint of future reuse to those generated by architecture modernization efforts, as RUP projects tend to yield artifacts that are more refined and more easily traceable. Another advantage of the RUP approach is that, normally, only artifacts of real value are produced. On the other hand, architecture modernization effort may result in the creation of scores of UML diagrams that have minimal practical value and dubious validation. In fact, this situation may negatively affect the acceptance of UML and compromise its future in the organization.

After UML has gotten traction in the organization through RUP-driven systems delivery projects, its use can be extended to enterprise architecture based on Zachman without running a risk of diminishing its appeal (see Figure 5).
This application has been attempted several times in the past, most recently in a context of Enterprise Unified Process (EUP).  

**Application 3: When starting from scratch, start with UML**

In my experience, efforts to introduce either RUP or Zachman to an organization are more likely to succeed when UML is already established. The graph in Figure 6 attempts to explain this by representing the dependencies among the three technologies.

As I've stated above, both Zachman and RUP depend on UML. RUP is exclusively UML-driven, whereas Zachman, being notation-agnostic, depends on a notation for its realization. UML is arguably the best notation to use with Zachman because it does not pose dependencies on other methods and, hence, can act as a point of departure into both RUP and Zachman. Moreover, even if you later decide not to use RUP or Zachman, UML remains a helpful and well-understood visual language that can be used for general analysis, for solutions analysis and design, and to improve team communication.

**Application 4: Learning RUP alongside Zachman (and vice versa) may be easier**

Learning RUP entails understanding how its structure and principles serve to guide project teams. The best ways to learn RUP, therefore, are through hands-on project experience or in a mentorship environment. To obtain this experience, taking on an active role during activities throughout the RUP lifecycle is imperative.

Such an approach definitely helps to gain appreciation for Zachman as well, as many aspects of Zachman have parallels within RUP practices. For example, the Zachman "function" aspect is addressed with RUP business process modeling, use cases, and sequence diagrams. Similarly,
"people" is handled by use cases and UI design, and "data" can be dealt with using class and object diagrams. Rows of the Zachman structure resonate with requirements-driven, incremental, and architecture-centric RUP principles, while columns correspond approximately to the objectives set for some UML models and best practices. For example, the "How" column has a strong process emphasis and can be compared with the UML Activity diagram.

Understanding Zachman is, arguably, comparatively easier than understanding RUP and UML because Zachman deals with static views of enterprise systems architecture rather than with dynamic models and processes. However, the path toward learning Zachman may strongly benefit from the application of some core RUP principles, such as "requirements-driven," "architecture-centric," "model-driven," and "iterative." My sense is that the Zachman Framework is easier to grasp when these principles infuse both the learning process and its practical application.

**Application 5: Zachman as an aid during RUP tailoring**

During RUP tailoring, work attention is focused on the structure of the development organization. Roles, time, and functions are important aspects to be considered during each such activity. Questions to be asked include: "Who in this process does architectural modeling?" "When does process modeling happen?" "What technique should be used for usage modeling?" and "How should responsibilities be divided based on level of detail?" With its rigid structure, Zachman may be just the right source of information to answer questions like these.

Although guidance for navigating workflows is included with RUP, it may be inapplicable or only a partial fit in real projects. To help process engineers deal with methodological variations, IBM Rational has created some specialized extensions to RUP, such as "RUP for COTS" (commercial off-the-shelf software) and "RUP for Systems Engineering." To pick an appropriate RUP variant as a baseline during tailoring work, a process architect must be cognizant of existing and future states of the organization's enterprise architecture. If the organization has used Zachman for enterprise architecture analysis/planning, the resulting framework can offer valuable clues to selecting the proper RUP variant. For example, concentration of artifacts in the Business Model (row 2) of the corresponding Zachman structure may imply emphasis on business modeling (see Figure 7) and thus suggest a closer look at the "RUP for Business Modeling" variant, whereas a higher artifact density in the Network column may hint at a possible role for "RUP for SOA" (service-oriented architecture).
In reality, it may happen that no artifacts describing organizational systems or no clear mapping with Zachman are available for various reasons. In such cases, the selection of a RUP variant can revolve around the anticipated distribution of artifacts created as an aid for project planning.

Application 6: After the RUP project is over, Zachman may take over

The activity of transitioning system artifacts from project to enterprise level (and vice versa) may be painful as it is not truly well-understood and has not been well-described. Although the RUP Deployment discipline is rich in activities supporting migration of a developed system into production, it scarcely covers activities surrounding the transition of architecture models created during a project. The same holds true for the intake side of RUP; there is no activity in inception or elaboration that systematically talks about using models available at the enterprise level.

Since the RUP lifecycle does not include the enterprise architecture discipline, in RUP, there is no guidance as to how models should be received by projects, transitioned back to the enterprise, and maintained after a system has been moved into production. This would have not been the case if enterprise architecture was formally acknowledged in the RUP planning and transition activities, which it is not.

This is one limitation enterprise and project architects have to work together on. Zachman has a role to play here -- as a framework for externalizing project architecture work. Although every organization will develop its own unique method for linking enterprise- and project-level artifacts,
a common goal is to link Zachman cells with RUP workflows and activities (see Figure 8). Unfortunately, these examples represent only part of the picture because they only describe the static artifact-to-cell mapping. Guidance for dynamic phase/iteration/activity events is not usually provided.

![Diagram](image)

**Figure 8: Artifact traceability between RUP to Zachman (illustration)**

When RUP project planning work is underway, it is desirable to introduce an easily navigable structure for linking artifacts. The most common way to do this is by replicating the RUP lifecycle structure. I have seen many examples of a setup in which folders correspond to iterations, with subfolders underneath organized by discipline. This approach has obvious drawbacks, as the project structure only matters while the project is still active and becomes irrelevant shortly after it is over.

A case for Zachman can be made at this point, as its core structure may be used for setting up project configuration in the source and document management tools. Simpler realizations can replicate Zachman structure using file system folders. If, at the conclusion of a RUP project, artifacts are moved into archives built to match the Zachman structure, they could be easily accessible by future enterprise and project teams. An investment such as this will benefit forthcoming RUP and enterprise projects and make the enterprise architecture more transparent and maintainable.

**Application 7: When planning a RUP project, look at the Zachman structure**

It is great news if the organization you work for already manages its artifacts in a structure organized according to Zachman. If not, it is still possible to benefit from methodically evaluating the Zachman structure during modeling activities and inquiring into aspects that the framework addresses. One thing to keep in mind about the framework is that it is a synergy of collective thinking by John Zachman and others gained through experiences with systems engineering projects. Different Zachman perspectives thus address many of the same issues that project teams may face (see Figure 9).
Some aspects of the Zachman structure are formulated as questions like what, where, and how, which are generic to all project types and may be translated into more project-specific questions. Others relate to important issues to be raised about the system architecture; for example "Rule Design" may lead to the "Will a solution require a rule engine?" question. Other statements like "Entity = Class of Business Thing" or "Process = Application Function" may convey the need for specific technologies, such as model-driven architecture (MDA) or business process modeling (BPM), which can be used at different stages in the development process.

Perspectives generated by Zachman may also aid during project and iteration planning. It is not hard to image how "assess iteration" or "develop RM plan" activities may make use of some of those nifty aspects.

Of course, for some (typically less experienced) architects, using Zachman the way I just described might sound too confusing, while seasoned professionals may find it redundant with their own grid of knowledge. I still believe that most practitioners will find Zachman to be a handy analysis reference during their work. And while setting up the environment for project planning is a non-trivial task in its own right, it is generally not difficult, and may prove very worthwhile for both the project and the organization.

**Application 8: Use RUP with Zachman to help bridge the gap between enterprise and project architecture**

A common trait of many developing organizations is to neglect enterprise architecture until complexity creep begins to be felt. Most organizations attempt to solve this problem by introducing an enterprise architecture practice and mandating its use to deal with cross-system and cross-project issues. This approach may help manage artifacts produced by projects, while at the same time reducing business risk by empowering projects with common models of the enterprise. Last but not least, an enterprise architecture practice will encourage interaction between enterprise and project teams, which will benefit both groups.
Quite often, a conflict occurs between enterprise and project architects regarding their respective spheres of influence. It all boils down to what artifacts (whether UML or not) best represent enterprise systems, what teams create them, and how they are maintained and used. The ultimate level of architecture transparency, represented in Figure 10, may be achieved when the lifecycle of an artifact can be clearly traced from a point of creation (during a development project or other enterprise project) all the way through whatever projects subsequently utilize the artifact, to a point where it is no longer required (possibly, after the system’s retirement). RUP and Zachman combined cover a significant part of the artifact lifecycle, which can help an organization realize the significant benefits of a unified, fully transparent architecture.

Figure 10: Cyclical lifecycle of an artifact

Summary

As leaders in their respective disciplines, UML, RUP, and the Zachman Framework can be used together in any organization to deliver better overall architectural value. Both RUP and Zachman are model-driven and require a notation to function. Since RUP dictates UML as a notation, it probably makes sense to standardize on UML for enterprise architecture as well, since there are generally no drawbacks to this.

Despite the fact that both RUP and Zachman rely on models, they have virtually no functional overlap. This is primarily because RUP is a process, while Zachman is a framework, but also reflects the fact that RUP targets project architecture while Zachman focuses on the enterprise organization.
Since both RUP and Zachman can depend on UML, the latter is the preferred methodology of the three to introduce first. Applying RUP to Zachman or vice versa makes for a better overall learning experience.

Using Zachman to classify available artifacts, or even just referencing the Zachman structure and principles, makes RUP tailoring easier as it provokes thinking about roles, artifacts, workflows, and activities essential to a development organization.

Project planning efforts also benefit from applying Zachman, as it can quickly lead you to artifacts that can be used during requirements gathering or analysis/design. Even when there are no artifacts linked to Zachman available, the Zachman structure itself is still very helpful because it provides various useful perspectives on the business problems a project addresses.

An organization will almost certainly benefit from supporting traceability of artifacts between enterprise architecture and its projects. This traceability can be gained by establishing control over the lifecycle of an artifact from the point of creation to retirement. Along the way, both RUP and Zachman can be applied to managing the artifact.

**A final thought**

When the time comes to create solutions that are flexible and maintainable, project and enterprise teams should learn to work together. Project people should understand the broader enterprise context, while their enterprise counterparts must constantly monitor projects to keep their knowledge base up-to-date. Applying joint use cases across RUP and Zachman may help narrow the gap between the enterprise and its projects, thus making the organization more efficient. In the end, that's what it's all about.

I would like to thank S. Ambler, D. Bentley, and J. Reading for their direct and indirect contributions to the content of this article.

**Notes**

1. Zachman and the Zachman Framework are short-hand ways of referring to the Zachman Framework for Enterprise Architecture. For a comprehensive description please visit [http://www.zifa.com](http://www.zifa.com)

2. *The Practical Guide to Enterprise Architecture* by James McGovern et al. is a good introduction to creating adaptive strategies to implementing EA.

3. The Zachman Framework strongly promotes the view that detail is the property of a cell, where detail means decomposition detail and not something that happens as you move from row to row.

4. An opinion is that the three middle rows of the Zachman structure (rows 2-5) are the abstractions of the rows 1 and 5. I also tend to think of the Zachman rows as having increasing levels of abstraction, moving from top to bottom.
5 Terry Moriarty, in the article "UML for the whole lifecycle," (see http://www.intelligententerprise.com/010416/products1_2.jhtml;jsessionid=K32N3LADW3WG0QSDLPSKH0CJUN2JVN) was the first to propose combining RUP, UML, and the Zachman Framework. Another good read on the correlation between enterprise architecture and other disciplines, including UML and Zachman, is by Scott W. Ambler at http://www.agiledata.org/essays/enterpriseArchitectureTechniques.html

6 In one of the latest revisions of RUP, the term "artifact" was changed to "work product."

7 Two good sources of information about RUP’s evolution are to be found at http://www.ibm.com/developerworks/rational/library/873.html and http://www.enterpriseunifiedprocess.com/essays/history.html

8 Although initially RUP only targeted software development, with the introduction of RMC, it now embraces other specializations as well.

9 The EUP Website (http://www.enterpriseunifiedprocess.com/) is a major source of information on enterprise architecture.


11 Noblestar ITIL Plug-in for RUP and RMC is a definite attempt to close this gap.

12 Several attempts at mapping RUP artifacts to Zachman structure have been made in the past. For examples of artifact mappings see http://www-128.ibm.com/developerworks/rational/library/372.html and http://www.enterpriseunifiedprocess.com/essays/zachmanFramework.html

© Copyright IBM Corporation 2006

Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)