Methodology Object Management

Charles W. Butler
Colorado State University

Over the past thirty years, many software development methodologies have emerged within the information technology profession. During this time period, no one methodology has dominated software projects as the best solution for development success. Rather than rejecting the value of any single software development methodology, Methodology Object Management is an enterprise meta development methodology that embraces software project planning, design, and deployment for numerous development methodologies including waterfall, prototyping, iterative, object-oriented, rapid application development, and agile techniques. Management Object Management's design utilizes the object-oriented paradigm to model classes and to implement processes (methods) and deliverables (data) for software development. Then, an instance of each software development methodologies is mapped to the MoM base class yielding a consistent, enterprise development architecture that can be measured and analyzed.

INTRODUCTION

Today, there are many software development methodologies used within most companies. Waterfall, prototyping, iterative, object-oriented, rapid application development, and agile techniques resonate through the industry. There are good reasons for all of these methodologies, and it is unreasonable to assume that any one of them will become a dominant methodology which will eliminate the others. Historically, software development methodologies have evolved from functional decomposition through structure analysis and design to information engineering, often leaving information technology (IT) professionals in a quandary regarding which methodology is best and which should be used for a software development project. Caught in the middle of this methodology debate is company management and customers whose expectations for successful application deployment often fall short. In fact, there are many software development approaches, and each has a legitimate role in the software community.

In practice, all software development methodologies, such as those cited above, have a legitimate role in many companies. Software requirements for diverse companies often dictate a development approach. For example, if a company's software systems are not object-oriented, then an alternate methodology would be better suited for its development projects. If a company's application portfolio is heavily invested in relational database technology, then a data modeling methodology would be a suitable one. Moreover, the decision to choose a specific development methodology is not always based upon identifiable criteria.

An enterprise approach for software development methodology should incorporate meta methodology, similar to that used in other IT functions or services. The IT profession has successfully implemented meta data concepts. Corporate data dictionaries often have meta data to provide high-level information about dictionary content. Large data warehouses can have meta data to catalog the organization and detail of warehouse data. Similar to meta data concepts that successfully guide data requirements, design, and deployment, a meta development methodology is needed to organize, manage, and control diverse development processes. In this article, Methodology Object Management (MoM) describes an enterprise meta methodology for software development. MoM provides company-wide management, execution, and control for software development projects. It is designed to be an integrated, tunable, configurable set of processes, deliverables, and control points that provide consistent development architecture for diverse software development methodologies.

METHODOLOGY OBJECT MANAGEMENT

MoM is an enterprise meta-methodology for planning, executing, and controlling a diverse software project portfolio utilizing multiple software development techniques. It is based on the object paradigm where a class is a template for defining the methods and variables for a particular type of object. Thus, methodology is a particular type of object. MoM uses a common framework for defining phases and tasks (methods) and deliverables (variables and attributes) for individual methodology instances. As part of an enterprise strategic for managing software development, a company can configure a consistent software development approach for multiple methodologies including waterfall, prototyping, iterative, object-oriented, rapid application development, and agile system development life cycles. MoM's design approach uses object oriented (OO) development and applies a common development architecture to different instances of software development methodologies. A development methodology framework is used to provide a consistent (common) structure for project management, operation, and control. Fundamentally, MoM uses a system development life cycle (SDLC) with a common work breakdown structure (WBS) to form a base class for development activities. This approach parallels OO's polymorphism (from the Greek meaning "having many forms") where a common interface generates different implementation schemes.

There are six important principles that underlie MoM's design. Each principle is briefly described below.

1. Granularity: Define a software development architecture that is simple, consistent, flexible, and expandable. Simplicity requires the number of project phases, phase activities, and deliverables be limited and consistent.
2. Phase content: Define development to be associated with five or six phases based upon industry practices and formulate the software engineering core development activities using industry best practices.
3. Task content: Restrict WBS tasks per phase to five or six common activities, providing consistent time recording within a company's time tracking system. An additional goal is to formulate the software engineering work requirements that produce accepted, useable deliverables promoting high quality systems and industry best practices.
4. Quality assurance: Identify quality assurance activities such as reviews and signoffs and associated compliance requirements that produce quality systems and support organizational and project management.

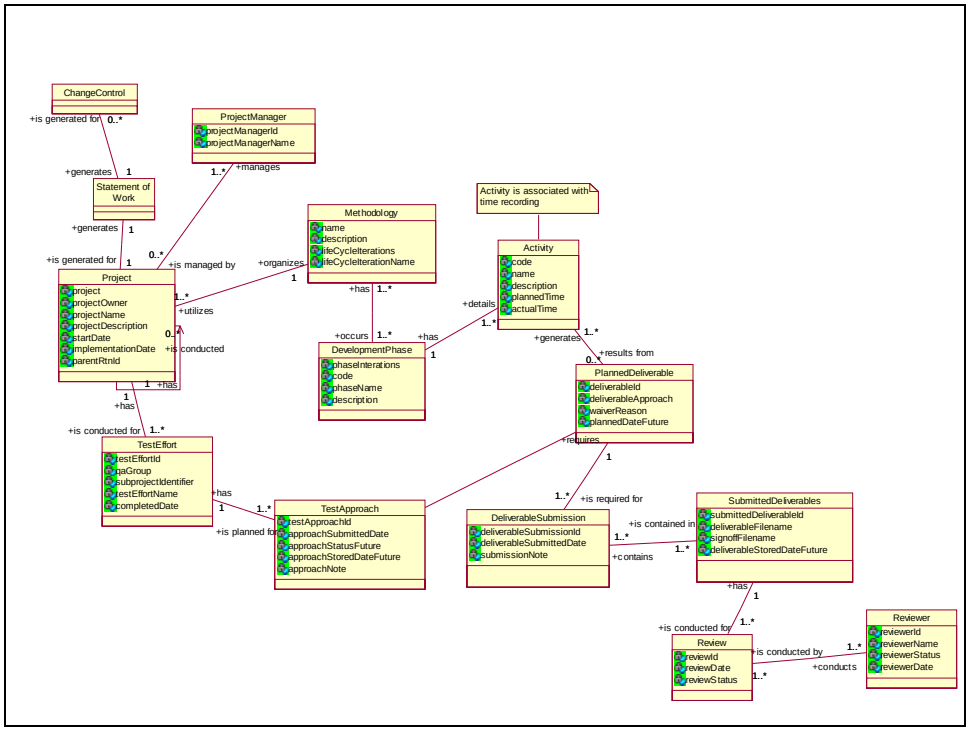
5. Reuse: Reapply development frameworks and development assets to existing as well as sunrise methodologies as they are introduced within a company.
6. Measurement: Measure development activities and provide management and customer accurate information regarding development costs. Using the MoM base class of development activities, management and reporting of development activities have a common framework for consistent measurement and reporting over time.

Another business concept, generally accepted accounting principles (GAAP), aids in describing MoM's design philosophy. GAAP's are successfully used to measure a company's financial worth. With GAAP's, a chart of accounts and consistent accounting rules are implemented to identify and value assets and liabilities. A parallel analogy is implemented in MoM. For example, the object oriented paradigm might produce a deliverable, use cases. In contrast, waterfall methodology might produce a deliverable, software requirements specification. While the approach and format between these two methodologies is different, the essence of both is the same, define what software requirements are needed for the system. Thus, MoM's design philosophy provides consistent methodology rules to identify and value development activity and deliverables.

Mom Class Diagram

In Figure 1, a class diagram for MoM is shown. There are five key classes in the class diagram including the Methodology, DevelopmentPhase, Activity, PlannedDeliverable, and Review classes. Using these classes, a base class is established upon which software

FIGURE 1
MoM CLASS DIAGRAM



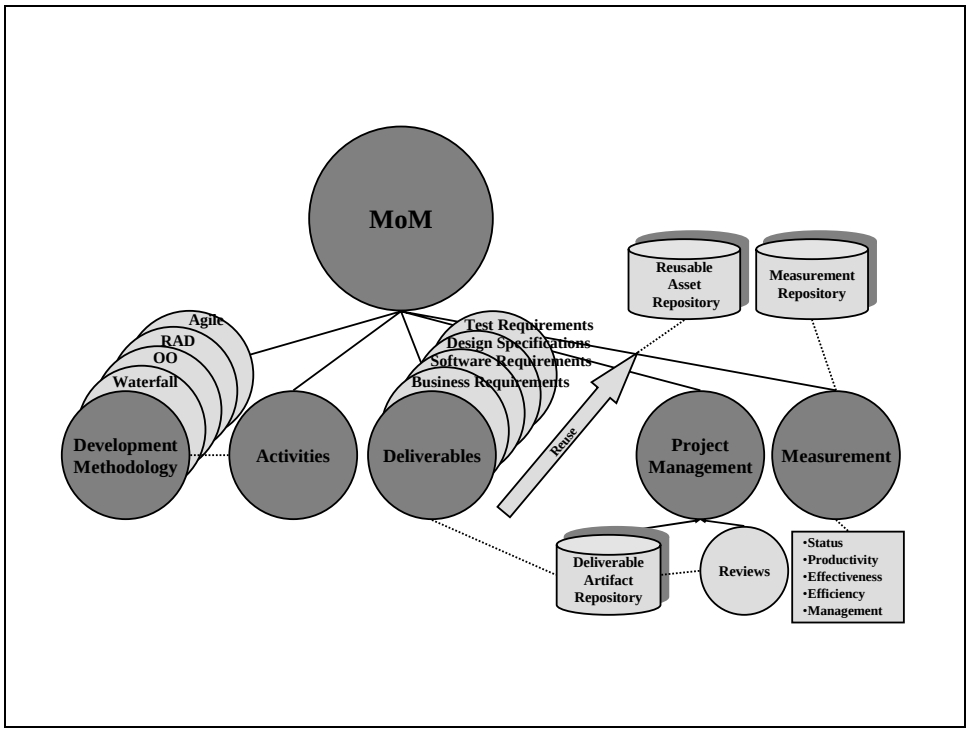
development is conducted to provide consistent structure and improved managerial and project reporting. For example, by instantiating the key classes, a common framework is defined for phases and tasks (methods) and deliverables for individual system development life approaches. With the association between the Methodology, DevelopmentPhase, and Activity classes, a work breakdown structure (WBS) of development activities binds different development activities to a common activity. The base class instance is defined and each discrete development methodology used in a company is instantiated against the base class. The MoM base class is the class upon which project management is conducted providing consistent structure and project reporting for development activities.

MoM Enterprise Framework

When instantiating several development methodologies, an enterprise framework defines the key elements required for successful software development. In Figure 2, the MoM enterprise framework identifies five key elements for successful implementation: development methodology, activities, deliverables, project management, and measurement. For each of these elements, MoM defines a base class for the following:

- Development methodology: Work activities and deliverables for system development life cycle instances such as waterfall, prototyping, iterative, object-oriented, rapid application development, and agile approaches.
- Activities: A standard WBS of development activities.

FIGURE 2
MoM ENTERPRISE FRAMEWORK



- Deliverables: Standard deliverables associated with best practices including business requirements, software requirements, design specifications, and test plans.
- Project management: Project management oversight (exclusive of project management phases and activities) including standard repositories for deliverables, consistent review gates for project activities, and compliance requirements to ensure adequate project control and quality deliverables.
- Measurement: Measures of project status, productivity, effectiveness, and efficiency that support management control and reporting objectives.

MoM DEVELOPMENT PHASES AND ACTIVITIES

MoM's common system phases establish the base class for the enterprise software development life cycle. These phases, representing a common system development life cycle, are identified and described in Table 1 and include project initiation, business requirements, analysis, design, build, test, and deployment phases.

TABLE 1
MoM BASE CLASS SDLC PHASES

PHASE CODE	PHASE NAME	PHASE DESCRIPTION
1	Project Initiation	Initial business study, develop initial scope, and identify business opportunities. A Statement of Work (SOW) and project scope is generated.
2	Business Requirements	Formulate the needed business functions and their associated interface and performance requirements.
3	Analysis	Completed system requirements outlining what the system will be. Identify the infrastructure, software, and data components.
4	Design	The definition of how the system will be implemented and how portions of the requirements are allocated to selected hardware processors and software components.
5	Build	Software is coded and unit tested and data base environments are populated. Infrastructure is constructed.
6	Test	Software and hardware components are system, integration and performance tested and readied for implementation.
7	Deployment	Newly developed system is placed into production and affected personnel are trained. Responsibility for the operation is transferred to the responsible organizational unit.

Base Class Activities

Within phases, MoM organizes the large number of development activities incorporated in various software development methodologies and their associated core and detailed activities. Using MoM, a common development activity structure is instantiated to establish consistent development activities. In order to accomplish this goal, a WBS of development activities (WDA) is defined. The WDA is a list of all MoM core development activities, similar to GAAP's chart of accounts. WDA numbering system is a convenient way to record development activity time and can be used for time tracking independent of various development

methodologies. Further explanation of MoM's chart of development activities is detailed in Table 2. In the WDA, key words are used to indicate responsibility. If the word, review, is used, then the role is participation. Another word, complete, indicates that the role is a producer of a deliverable. Using the WDA, an instance of development activities is dynamically allocated (for a project) to manage development within existing a company's inventory of existing software development methodologies.

TABLE 2
MoM BASE CLASS WDA

ACTIVITY CODE	ACTIVITY NAME	ACTIVITY DESCRIPTION
010	Conduct General Project Activity	Participate in project activity not generally associated with the following activities
020	Complete Business Requirements	Construct and review the business requirements including functional requirements as input for completing and improving development sizing.
030	Complete Development Sizing	Define the proposed development scope and size the development effort.
040	Complete Project Plan	Identify development activities within a project plan and ensure an understanding of project plan's impact on development activities.
050	Complete Test Strategy	Define and review the proposed high level test approach, test scope and sizes the test effort including studying and reviewing the business requirements and system requirements
060	Review Project Plan	Participate in the review to ensure accuracy and correctness of the project plan and scope.
070	Complete System Requirements	Construct the system requirements including infrastructure, software, data, qualification and performance requirements.
080	Complete Development Environment	Determine the development environment for the project including automated tools.
090	Review System Requirements	Participate in a review to ensure accuracy and correctness of the system requirements.
100	Complete Test Plan	Construct and review the test plan defining what testing is to be conducted. Activity includes studying and reviewing the system requirements as critical input for completing and improving the test plan and the test environment.
110	Complete Preliminary Design Specification	Construct the preliminary design specifications for how the system will be implemented.
120	Review Preliminary Design	Participates in a review to ensure the accuracy and correctness of the preliminary design and qualification.
130	Complete Detailed Design Specifications	Construct the detailed design specifications for how the system will be implemented.

TABLE 2
MoM BASE CLASS WDA

ACTIVITY CODE	ACTIVITY NAME	ACTIVITY DESCRIPTION
140	Review Detailed Design	Participates in a review to ensure the accuracy and correctness of the detailed design and qualification.
150	Complete Test Design	Construct and review the test design defining how testing is to be conducted including test cases and data for the proposed test plan. QA determines the test environment for the proposed test plan including automated tools.
160	Build Units and Components	Generates code and component units; generate data components including data bases.
170	Execute Unit Tests	Execute low level testing including unit and low level integration testing.
180	Review Unit Test	Review unit level development and qualification in order to ensure an adequate basis for system testing.
190	Conduct Test Preparation	Determine the test environment for the project including automated tools.
200	Execute System Test	Execute the qualification plan.
210	Complete System Test Findings	Construct and review test execution results providing input into the decision to deploy by defining business risks associated the test activities and their results.
220	Manage System Test Assets	Store, classify, and share test assets including test cases, test scenarios, and test data; promote reuse and reduce redundancy of test assets.
230	Complete Development Evaluation	Studies the development and test effort and identify strengths and weaknesses and formulates recommendations
240	Promote Metrics	Populate common metrics gathered for a project into a management summary. The evaluation includes both development and test activities.

Why is the WDA so important? A review of one company's project management climate provides valuable insight into the WDA's critical role. In this company, executive management asked two questions. "What is the cost of development?" and "What is the cost of testing?" In order to answer these questions, initiatives were taken to quantify professional staff time and other resources associated with development and testing. Testing was targeted first, since it is a subset of a software development project. Queries were executed against the company's time tracking system. For the targeted fiscal year, there were no standard, consistent tasks that the quality assurance staff used for time recording (such as the WDA). Further analysis revealed that no one actually knew who made up the quality assurance staff. In an effort to answer the second question, a list of "quality assurance staff" was drafted and queries were executed, using the list as the query selection criteria, to extract time for activities against which these identified staff members recorded time. It was assumed that their time was test oriented. This effort uncovered

over 1500 tasks used for time recording. For example, there were tasks named execute test, execute tests, and test execution. As similar approach for developers uncovered over 4000 tasks used for time recording. The point is that the WDA provides a consistent architecture for software development activities and project organization.

AN OBJECT ORIENTED CLASS INSTANCE

To illustrate MoM meta methodology, object oriented methodology is used. In today's software development world, projects that are OO generally use IBM's Rational Unified Process (RUP). RUP is an iterative methodology comprising of four distinct phases: inception, elaboration, construction, and transition. Within each RUP phase, multiple iterations are supported. Each iteration progresses through six key workflows: business modeling, requirements, analysis & design, implementation, test, and deployment. Table 3 is an example of RUP as a MoM instance. This table contains the mapping of a sample RUP project to the MoM class diagram. For the Methodology class, the attribute values for name, lifeCycleIterations, and lifeCycleIterationName are RUP, 2, and Elaboration, respectively. For the DevelopmentPhase class, the phases are RUP's work flows, Requirements, Analysis & Design, Implementation, Test, and Deployment. Finally, for the Activity class, RUP's core activities map to MoM's activities, as represented by each table row. For the Activity class, there is often a one to many multiplicity of a methodology's activities to a MoM activity. This multiplicity results in a simplified MoM chart of accounts. As shown in Table 3, an object oriented project can be mapped to MoM's standard architecture.

TABLE 3
MoM WDA MAPPING WITH RUP CORE ACTIVITIES

LIFE CYCLE PHASE ITERATION: ELABORATION - PHASE ITERATION: 3				
MoM			**RUP**	
PHASE	**#**	**ACTIVITY**	**WORK FLOW**	**CORE ACTIVITY**
Project Initiation	020	• Complete Business Requirements	Business Modeling	•Develop a Domain Model
Business Requirements	050	• Complete Test Strategy	Requirement	•Define Test Scope, Resource Requirements, and Risks
Analysis	070	• Complete System Requirements	Requirements	•Refine the System Definition
	090	• Conduct System Requirements Review	Requirements	•Refine the System Definition
	100	• Complete Test Plan	Test	•Plan Test
Design	110	• Complete Preliminary Design Specification	Analysis & Design	•Refine Architecture
	120			
	130	• Conduct Preliminary	Analysis & Design	•Refine Architecture

TABLE 3
MoM WDA MAPPING WITH RUP CORE ACTIVITIES

LIFE CYCLE PHASE ITERATION: ELABORATION - PHASE ITERATION: 3				
MoM			**RUP**	
PHASE	**#**	**ACTIVITY**	**WORK FLOW**	**CORE ACTIVITY**
	140 150	Design Review • Complete Detailed Design Specifications • Review Detailed Design Specifications • Complete Test Design	Analysis & Design Analysis & Design Test	•Design Components and Design Database •Design Components and Design Database •Design Test
Build	160	• Build Units and Components	Implementation	•Implement Components
Test	200 210 220	• Execute System Test • Complete System Test Findings • Manage System Test Assets	Test Test Test	•Implement Tests in System Test Stage •Execute Tests in Integration Test Stage •Evaluate Test
Deployment	230	• Complete Development Evaluation	Deployment	•Manage Process Improvement

BUSINESS INTELLIGENCE

MoM concepts, such as the WDA, can be expanded to provide more intelligence, providing management with measurements for work being conducted and who conducts work. One extension identifies the participant as project manager, developer, tester, or customer. Table 4 outlines participation extensions. These extensions can be used to record time accurately, since multiple participants contribute to a task activity. For example, consider the Design phase and Review Detailed Design Specifications activity in Table 3. Both developers and testers could participate in this activity. So, the phase-activity code can be concatenated with a role code when recording time. In this example, the code, 41403 (4 – Design phase; 140 – Review Detailed Design Specifications; 3 – tester), records in time tracking software a tester's time spent completing Conduct Detailed Design Review activity during the Design phase.

TABLE 4
ROLE AND PARTICIPATION WDA EXTENSIONS

ROLE CODE	**ROLE**
1	Project manager
2	Developer
3	Tester
4	Customer

Appending the core activity identifier can also account for RUP methodology uniqueness and iterative methodology activity. In Table 3, the RUP project is in the third iteration of the elaboration phase. The MoM development activity can be coded representing the phase and activity codes The participant is a tester. So, a core development activity, 62003, represents development work for the Execute System Test activity (200) in the Test phase (6). Further, WDA accounts for RUP iteration using life cycle phase and phase iteration attributes. The phase and iteration attributes, lifeCycleIterations and phaseIterations, are included in the class model in the Methodology and Phase classes. The life cycle phase attribute is the number of time that a life cycle is repeated (In RUP, this attribute represents inception - 1, elaboration - 2, construction - 3, and transition - 4). The phase iteration attribute is the number of times a life cycle phase is repeated (In RUP, this attribute represents the number of iterations within RUP phases). In the example cited above (the third iteration within the elaboration phase), the extended WDA code accounts for more project information (when and who) by recording tester's time as 6200323. Thus, a general MoM WDA coding convention is defined in the following table.

TABLE 5
MoM ACTIVITY CODE CONVENTIONS

Bytes	**Description**
1	Phase
2-4	Activity
5	Participant
6	Life Cycle Phase Iteration
7	Phase Iteration

While the goal of this article is to describe MoM meta methodology, all MoM concepts are not detailed. However, other MoM elements, including standard deliverables, project management, and measurement, are briefly described in the following section.

ADDITIONAL MoM ELEMENTS

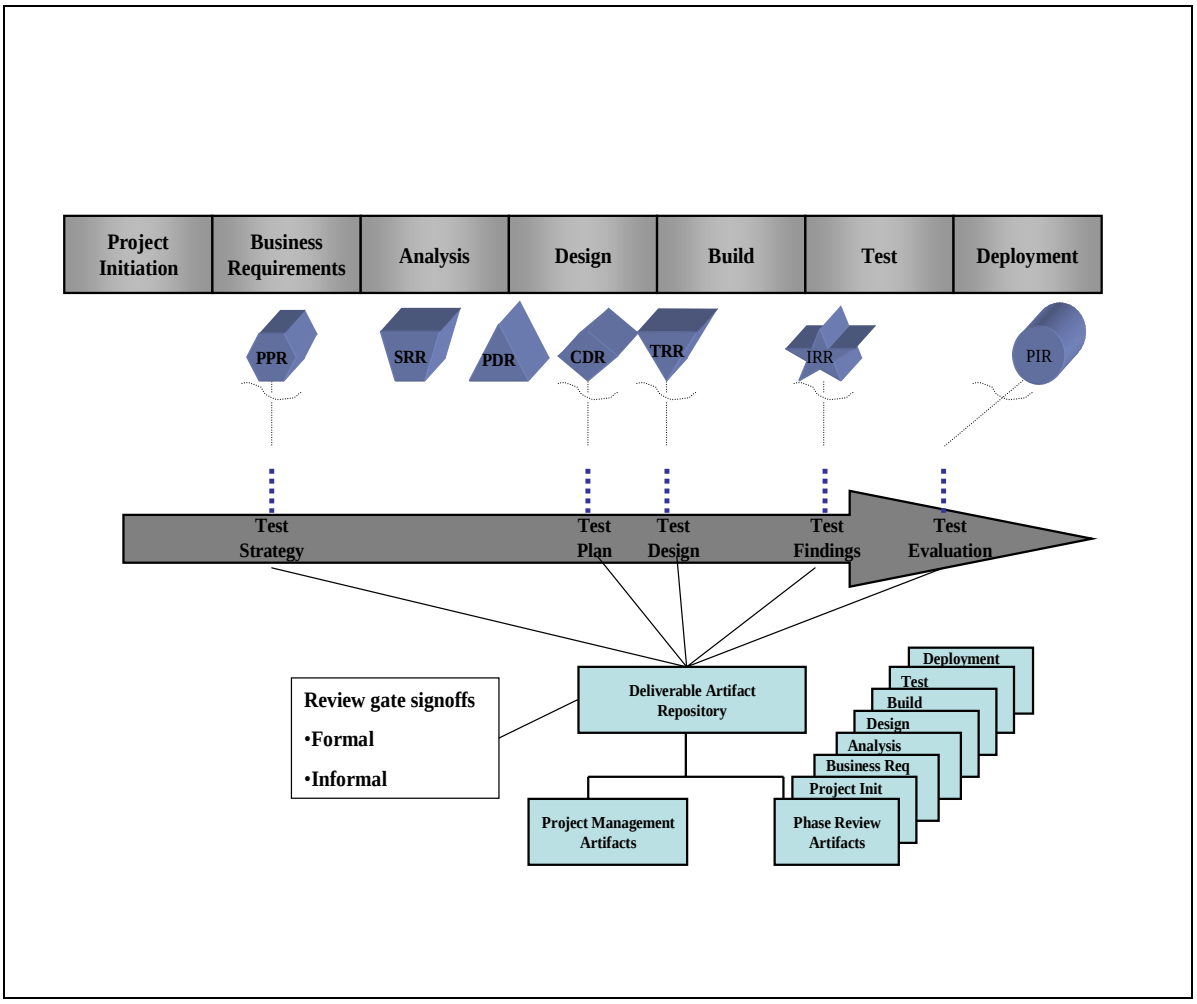
Figure 3 illustrates several other important MoM elements. In order to implement success software development, the MoM base class includes standard instances for deliverables and artifact storage, project management reviews and measurement.

- Base class deliverables and artifact storage: Key artifacts associated with major development phases include four major deliverables, business requirements, software requirements, design specification, and test plans. For example, as shown in Figure 3, a waterfall methodology's quality assurance processes can result in several deliverables such as a test strategy, test plan, test design, test findings, and test evaluation. These artifacts should be stored in standard repositories that correspond to the Deliverable Artifact Repository of MoM base class.
- Project management and compliance: Project management components include artifact repositories, quality reviews, and signoffs for compliance. The MoM base class instance includes the review gates identified in Figure 3, Project Plan Review (PPR), Software Requirements Review (SRR), Preliminary Design Review (PDR), Critical Design Review (CDR), Test Readiness Review (TRR), Implementation Readiness Review (IRR), and Post Implementation Review (PIR). Using RUP methodology, there are phase

milestones such as the Operational Readiness Milestone and the Release Milestone. The base class review gates, IRR and PIR, map to these milestones. When reviews are conducted, MoM processes require signoff procedures. In Figure 3, both informal and formal signoffs are instances of compliance. Regardless of the signoff format, compliance requires documented signoffs be stored in the Deliverable Artifact Repository.

- Measurement: Successful quality assurance processes embrace measurement to provide accurate statements of performance and to maintain control. Measurements can also provide compliance information. Consider a waterfall project where five review points are planned, PPR, SRR, CDR, TRR, and IRR. Meta methodology requires that these gate reviews be conducted, signoffs be generated, and artifact stored. If only four of the review gates are conducted, then management is informed that 80% of the gates were successfully completed. If only three of the major deliverables are completed, reviewed, and stored in standard artifact repositories, then these measurements can be generated for management review and evaluation.

FIGURE 3
ADDITIONAL MoM ELEMENTS



IN REVIEW

With increased complexity associated with software development, a meta methodology is needed to integrate the various development approaches utilized in the IT profession. Methodology Object Management's overall goal is to embrace discrete development approaches under a single base class for standardization, consistency, and dynamic management. MoM is designed to be simple and reduces existing focus on an unmanageable volume of inputs, tasks, outputs, and deliverables. It also uses consistent test activities and deliverables so that high priority activities can be established. Just as important, MoM is configurable using the object oriented development polymorphism concept that supports dynamic binding.

In MoM, no single development methodology is mandated. Rather, it maps to existing waterfall, prototyping, iterative, object-oriented, rapid application development, and agile life cycles. Since its design basis used object-oriented technology, it is readily expandable. It can extend to sunrise methodologies, embrace extensions to existing methodologies, and gracefully retire sunset methodologies. MoM even promotes OO's reuse strategy. So, instead of debating the virtues of a single development methodology, MoM embraces the methodology community by implementing class instances for various software development techniques. Dynamic binding of unique methodology concepts supports configurable approaches within a company. A critical outcome using MoM is a consistent view of all projects regardless of the project's development methodology. A building block approach for deliverables includes an artifact repository for shared deliverable components. The time has arrived for meta methodology technology to overcome the ongoing issues associated with disparate software development life cycles and the high cost of uniqueness among today's solution.

REFERENCES

Amber, Scott W. (2002). Agile Modeling. New York. Wiley Computer Publishing.

Cohan, Mike (2004). User Stories Applied: For Agile Software Development. New York, Addison-Wesley.

Coad, Peter and Yourdon, Edward (1991). Object-Oriented Analysis. New Jersey, Prentice-Hall, Inc.

Pressman, Roger S (2001). Software Engineering A Practitioner's Approach. New York, McGraw-Hill Companies.

Rumbaugh, James, Jacobson, Ivar, and Booch, Grady (1999). The Unified Modeling Language Reference Manual. New York, Addison-Wesley.