# EVALUATION & VALIDATION OF WORK PRODUCTS IN UNIFIED SOFTWARE DEVELOPMENT PROCESS

Meena Sharma[1] and Rajeev G Vishwakarma[2]

[1]Department of Computer Engineering, IET,
Devi Ahilya University, Indore, India

meena@myself.com

[2]Department of Computer Engineering & Science, SVITS,
Rajiv Gandhi Technical University of MP, Indore, India

rajeev@mail.com

## *ABSTRACT*

*A work product is a general abstraction that represents something obtained from the software development process. A work product may have many work product kinds. For instance, we might want to have a series of work product kinds that keeps up a correspondence to the overall intent of work products, such as specification, plan, or model. There are three types of work products- Artifact, Outcome and Deliverable. An artifact is formal work product that is produced, modified, or used by a task, defines an area of responsibility and is subject to version control. Outcome primarily describes intangible work products that are a result or state. An Outcome can also be used to represent an informal work product. A deliverable is an output from a process that has a value, material or otherwise, to a customer or other stakeholder. In this paper we have investigated the software metrics to evaluate different work products in analysis & design. Tasks have input and output work products. Roles use work products to perform tasks, and produce other work products in the course of performing tasks. Work products are the responsibility of a single role, making responsibility easy to identify and understand, and promoting the idea that every piece of information produced in the process requires the appropriate set of skills. Even though one role may "own" the work product, other roles will use the work product, perhaps even updating it if the role has been given permission to do so. We have suggested some metrics pertaining to the input and output artifacts. The metrics that we developed are for analysis and design process, software architecture document artifact, and design model artifact. Also we investigated how to quantify the artifact checklist items and make a decision about the quality for different attributes of the process and artifacts, and finally deciding upon the overall quality.*

## *KEYWORDS*

*Work Product, Artifact, Analysis & Design, Review Technique, Metrics, Checklist, Role*

## 1. INTRODUCTION

Artifacts based metrics are the software metrics that are developed by us from the work products point of view. The work is based on developing some checklist based metrics for the artifacts developed during the analysis & design (A&D). We have developed metrics using checklist approach to satisfy quality from CMMI [West,2004] perspective also. In unified process the artifacts that are produced; are Software Architecture Document (SAD), Design Model, Analysis

Model, Deployment Model and the Data model. In the following section we emphasize on the checklists based metrics for some of these artifacts [Chrissis,2006] [Ahern,2005].
We have developed metrics for A&D Process, Artifact Software Architecture Docu

ment, Artifact Design Model, Artifact Deployment Model and Artifact Data Model [Sharma,2009]. Looking in to the space limitation we are describing process metrics for A&D and artifact metrics for Software Architecture Document and Design Model. FI/PI/NI/NA is the abbreviation as described below. We have taken the idea of FI/PI/NI/NA from Capability Maturity Model Integration (CMMI) Artifacts are evaluated on the basis of FI/PI/NI/NA. This is shown in Table 1.

Table 1. Explanation of evaluation by the auditor

| | |
|---|---|
| FI | Fully Implemented in compliance with the documentation |
| PI | Partially Implemented in compliance with the documentation. One or more noncompliance's noted. |
| NI | Not Implemented, not even partly in compliance with the documentation. One or more noncompliances noted. |
| NA | Not Applicable for this project. Approved waivers exist. |

## 2. INSPECTIONS, REVIEWS & WALKTHROUGHS

The IEEE standard Glossary defines these three kinds of review and evaluation efforts as:

### 2.1. Review

A formal meeting at which an work product, or set of work products are presented to the user, customer, or other interested party for comments and approval.

### 2.2. Inspection

A formal evaluation method in which work products are examined in full legth by a team member or group other than the author to identify and discover errors, violations of development standards, and other problems.

### 2.3. Walkthrough

A review procedure in which a developer leads one or more members of the software development team during a segment of a work product that she has documented while the other members ask questions and make comments about technique, style, possible errors, violation of development standards, and other problems. In the most common and usual form of term, a walkthrough is step by step simulation of the execution of a procedure, as when walking through code line by line, with an imagined set of inputs. The term has been extended to the review of material that is not procedural, such as data descriptions, reference manuals, specifications, etc. The rules governing a walkthrough are:

- Provide adequate time
- Use multiple sessions when necessary
- Prepare a set of test cases
- Provide a copy of the program being tested to each team member
- Provide other support materials

Both Walkthrough and Inspection require preparation and study by the team members, and scheduling and coordination by the team moderator. Inspection involves a step-by-step reading of the product, with each step checked against a predetermined list of criteria. These criteria include checks for historically common errors. The participants in Walkthroughs may include Walk-through leader, Recorder, Author or Team member. The participants in Inspections may include Inspection leader, Recorder, Reader, Author or Inspector.

Walkthroughs vary from inspections in that the software engineer does not narrate a reading of the product by the team, but provides test data and leads the team for a manual simulation of the system. The test data is walked through the system, with intermediate results kept on a blackboard or paper. The test data should be kept simple given the constraints of human simulation. The purpose of the walkthrough is to encourage discussion, not just to complete the system simulation on the test data. The walkthrough and inspection procedures should be performed on the code produced during the construction stage. Each module should be analyzed separately and as integrated parts of the finished software. Design reviews and audits are commonly performed as stages in software development as follows: System Requirements Review, System Design Review, Preliminary Design Review, Final Design Review, and Final Review.

The suggested metrics can be executed making use of all the modes of evaluation and roles involved. So we can say that Inspections, Reviews, and Walkthroughs can be performed by the roles specified, making use of the proposed metrics. As a result we have better quality artifacts leading to good quality software and high rate of customer satisfaction. This means win-win situation for all the stakeholders. The main roles involved are Moderator, Recorder, Reviewer, Reader, and Producer.

## 3. ROLES INVOLVED IN EVALUATION

In a *review,* a work product is examined for defects by individuals other than the person who produced it. A work product is any important deliverable created during the requirements, design, coding, or testing phase of software development.

Research shows that reviews are one of the best ways to ensure quality requirements, giving you as high as a 10 to 1 return on investment. Reviews help you to discover defects and to ensure product compliance to specifications, standards or regulations. Software *Inspections* are a disciplined engineering practice for detecting and correcting defects in software artifacts, and preventing their leakage into field operations.

Inspections, reviews and walkthroughs are a reasoning activity performed by practitioners playing the defined roles of *Moderator, Recorder, Reviewer, Reader, and Producer.*

### 3.1. Moderator

Moderator is responsible for ensuring that the inspection procedures are performed through out the entire inspection process. This is the leader of the inspection. The moderator plans the inspection and coordinates it. The responsibilities include

- Verifying the work products readiness for inspection
- Verifying that the entry criteria is met
- Assembling an effective inspection team
- Keeping the inspection meeting on track
- Verifying that the exiting criteria is met

### 3.2. Recorder

The Recorder will document all defects that arise from the inspection meeting. This documentation will include where the defects was found. Additionally, every defect is assigned a defect category and type. This role is also known as 'Scribe'.

### 3.3 Reviewer

All of the Inspection Team individuals are also considered to play the Reviewer role, independent of other roles assigned. The Inspector role is responsible for analyzing and detecting defects within the work product. This role is also known as 'Inspector'.

### 3.4. Reader

The reader is responsible for leading the Inspection Team through the inspection meeting by reading aloud small logical units, paraphrasing where appropriate. She is the the person reading through the documents, one item at a time. The other inspectors then point out defects.

### 3.5. Producer

She is the person who originally constructed the work product. The individual that assumes the role of Producer will be ultimately responsible for updating the work product after the inspection. This role is also known as 'Author'. She is the person who created the work product being inspected. In an evaluation mode, the producer describes the product and asks for comments from the participants. These gatherings generally serve to inform participants about the product rather than correct it.

## 4. WORK PRODUCT AND CHARACTERISTICS

The work products are documents, models, or model elements. The models are collections of like things (the model elements) so the recommended metrics are listed here with the models to which they apply: it is usually obvious if a metric applies to the model as a whole, or an element. Explanatory text is provided where this is not clear.

### 4.1. Work Product Characteristics

In general, the characteristics we are interested in measuring are the following:

- Size - a measure of the number of things in a model, the length of something, the extent or mass of something
- Quality
- Defects - indications that a work product does not perform as specified or is not compliant with its specification, or has other undesirable characteristics
- Complexity - a measure of the intricacy of a structure or algorithm: the greater the complexity, the more difficult a structure is to understand and modify, and there is evidence that complex structures are more likely to fail
- Coupling - a measure of the how extensively elements of a system are interconnected
- Cohesion - a measure of how well an element or component meets the requirement of having a single, well-defined, purpose
- Primitiveness - the degree to which operations or methods of a class can be composed from others offered by the class
- Completeness - a measure of the extent to which a work product meets all requirements (stated and implied-the Project Manager should strive to make explicit as much as

possible, to limit the risk of unfulfilled expectations). We have not chosen here to distinguish between sufficient and complete.
- Traceability - an indication that the requirements at one level are being satisfied by work products at a lower level, and, looking the other way, that a work product at any level has a reason to exist
- Volatility - the degree of change or inconclusiveness in a work product because of defects or changing requirements
- Effort - a measure of the work (staff-time units) that is required to produce a work product

Not all of these characteristics apply to all work products: the relevant ones are elaborated with the particular work product in the following tables. Where several metrics are listed against a characteristic, all are potentially of interest, because they give a complete description of the characteristic from several viewpoints. For example, when considering the traceability of Use Cases, ultimately all have to be traceable to a (tested) implementation model: in the interim, it will still be of interest to the Project Manager to know how many Use Cases can be traced to the Analysis Model, as a measure of progress.

## 4.2. Artifacts

Artifacts are tangible well-defined work products consumed, produced, or modified by tasks. Artifacts may be composed of other artifacts. For example, a model artifact can be composed of model elements, which are also artifacts. They may serve as a basis for defining Reusable Assets. Roles use artifacts to perform tasks and produce artifacts in the course of performing tasks.

Artifacts are the responsibility of a single role, making responsibility easy to identify and understand, and promoting the idea that every piece of information produced in the method requires the appropriate set of skills. Even though one role might "own" a specific type of artifact, other roles can still use the artifacts, and perhaps even update them if the role has been given permission to do so.

Artifacts are generally *not* documents. Many methods have an excessive focus on documents, and in particular on *paper documentation*. The most efficient and pragmatic approach to managing project artifacts is to maintain them *within* the appropriate tool used to create and manage them. When necessary, one may generate documents (snapshots) from these tools, on a just-in-time basis.

Examples artifacts:

- A use case specification stored in Microsoft® Word®
- A design model stored in Rational Software Architect.
- A project plan stored in Microsoft® Project®.
- A defect stored in Rational ClearQuest.
- A project requirements database in Rational RequisitePro.

Note also that formats such as on whiteboards or flip charts can be used to capture pictorial information such as UML diagrams, tabular information such as short lists of status information or even textual information such as short vision statements. These formats work well for smaller, collocated teams where all team members have ready access to these resources.

However, there are still artifacts which either have to be or are best suited to being plain text documents, as in the case of external input to the project, or in some cases where it is simply the best means of presenting descriptive information. Where possible, teams should consider using

collaborative Work Group tools, such as WikiWiki webs or Groove to capture textual documentation electronically, thus simplifying ongoing content and version management. This is especially of importance where historic records must be maintained for purposes such as fulfilling audit requirements. For any nontrivial development effort, especially where large development teams are involved, work products are most likely to be subject to version control and configuration management. This is sometimes only achieved by versioning the container work product, when it is not possible to do it for the elementary, contained work products. For example, in software development, you may control the versions of a whole design model, or design package, and not the individual classes they contain.

## 5. UNIFIED PROCESS AND PROCESS CHECKLIST

The unified process expresses the A&D process in terms of roles, artifacts, activities, and workflow. Roles perform the activities as per the workflow and produce the artifacts. In order to produce artifacts the activities need input artifacts also. The metrics we are developing may be applied to input artifacts or output artifacts. These artifacts are the part of unified software development process. Now we describe the general process checklist items as below for the artifacts. Note that these are process perspective only. All the artifacts must be evaluated against each metrics. First we provide the details pertaining to project as shown in Table 2.

**Table 2.** Project Details.

| **Project Details Metrics** | |
|---|---|
| 1 | Project Name: Write the title of the project |
| 2 | Project Phase and Iteration: Which phase and iteration the project is running |
| 3 | Project Manager: Name of the Project Manager |
| 4 | Owner/Author: Owner of the artifact or the author of the artifact |
| 5 | Date of Audit: Mention the date of the audit |
| 6 | Auditor: Write the name of the auditor |
| 7 | Audit Effort (hours): Number of hours taken up to conduct the audit |

All the checklist items must be satisfied for the attributes. These checklists are also called as Process Evaluation Checklist (PEC). The objective performing the evaluation using the checklist for different artifacts is to reduce failures in the production of artifacts in unified software development process. We evaluate in the form of a checklist for different categories as shown in Table 3 and Table 4. There are two types of process checklists. There are general and specific checklists. Table 3 shows general process checklist while Table 4 shows the A&D process specific checklist. Checklist approach is one of the important factors to develop the metrics and the quality model in CMMI [Burwick,2008]. Now we present the questionnaire of the checklist metrics for different checklist items in Table 3 for different categories.

**Table 3.** General Checklist Items for Process

| **General Process Checklist Items** | |
|---|---|
| **People & Training** | |
| 1 | Were non-staff resources (equipment, facilities, and tools) made available for the project's process activities? |
| 2 | Was the assigned staff formally trained in how to perform the process activities (including tool training, if needed)? |
| **Document Control** | |
| 3 | Are artifacts, meeting records, other documentation, etc. produced by this process under the defined configuration management for this project? |

| **Stakeholder Involvement** | |
|---|---|
| 4 | Do records exist that demonstrate stakeholder participation in all reviews including decision points? |
| **Measuring process effectiveness and efficiency** | |
| 5 | Have measures demonstrating process execution been collected throughout the project? |
| **Process Evaluation** | |
| 6 | Have PPQA audits been performed as scheduled? |
| **Working with Management** | |
| 7 | Do meeting records exist that demonstrate review by project management in accordance with the projects schedule? |
| **Process Improvement** | |
| 8 | Has a lessons learned document been created for this process and submitted to the process engineering group? |
| 9 | Have any process change requests been generated from the execution of this process? |

Table 4 shows the process specific checklist for analysis and design. This checklist will evaluate the process performed during analysis and design workflow.

**Table 4.** A&D Process Specific Checklist Items

| **Process Specific Checklist Items** | |
|---|---|
| 1 | Has Candidate Architecture been identified? |
| 2 | Has alternate solutions been applied in arriving at the Architecture |
| 3 | Have Use Case Realizations been created? |
| 4 | Has the Analysis Model been created? |
| 5 | Has the Design Model been created? |
| 6 | Has the Deployment Model been created? |
| 7 | Has the Software Architecture Document been updated with previous information? |
| 8 | Has preliminary Use Case Analysis been performed? |
| 9 | Have architecturally significant Analysis Classes been described? |
| 10 | Has behavior analysis been performed? |
| 11 | Has detailed Use Case Analysis been performed? (refinement of analysis classes and use case realizations) |
| 12 | Have design elements been identified in the Design Model? |
| 13 | Has behavior analysis been reviewed? |
| 14 | Has Component Design been performed? |
| 15 | Has Use Case Design been performed? |
| 16 | Has Subsystem Design been performed to generate Design Subsystem part of the Design Model? |
| 17 | Has Class Design been performed to generate Class Design in the Design Model? |
| 18 | Have Test Class and Test Packages been designed? |
| 19 | Has Database Design been performed? (optional - evidence in Data Model) |
| 20 | Has Component Design been Reviewed? |
| 21 | Has the architecture been refined? |
| 22 | Have the Design Mechanisms been identified and documented in the Software Architecture Document? |
| 23 | Have the Design Elements been identified and documented in the Software |

| | |
|---|---|
| | Architecture Document? |
| 24 | Have existing Design Elements been evaluated to be incorporated? |
| 25 | Have the Run-Time Architecture been described in the Software Architecture Document? |
| 26 | Has the distribution been described in the Software Architecture Document? |
| 27 | Has the Architecture been reviewed? |

This should be noted that the project details, general process checklist items and specific process metrics as shown in Table 2, Table 3 and Table 4 will be applicable to all the artifacts. So these process metrics must be repeated for each and every artifact in the iteration. We will not describe these items again and again for every artifact, but we must make sure that project details, general process metrics and specific process metrics must be followed for every artifact. After describing process metrics we describe artifact metrics in the following sections.

## 6. METRICS FOR SOFTWARE ARCHITECTURE DOCUMENT

This artifact is also termed as SAD. It offers a complete and comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It provides as a communication medium between the software architect and other team members of the project regarding architecturally significant decisions and documents that have been prepared on the project. The metrics for Software Architecture Document is as follows, shown in Table 5 and Table 6. In Table 5 describes general artifact checklist and Table 6 depicts specific Software Architecture Document metrics. The general artifact checklist will also apply to all the artifacts so it will not be described for rest of the artifacts hereon.

**Table 5.** General Artifact Checklist- Applicable to All Artifacts

| | |
|---|---|
| **Corrective Action Efficiency** | |
| 1 | Have corrective actions carried out on previous noncompliance prevented recurrence of the noncompliance? That means we have to evaluate the noncompliance and its remedies done earlier. |
| **Correct Template** | |
| 2 | Does the layout correspond to the template defined for this type of artifact? |
| **Revision History** | |
| 3 | Has the artifact revision history been maintained with a description for each of the major changes including the reason for the changes? |
| **Review cycle** | |
| 4 | Was the audited version of the document managed through the review process as defined in its Project Plan? |
| **Version Numbering** | |
| 5 | Was the version numbering used in the artifact? |
| **Production Frequency** | |
| 6 | Was the artifact produced with the defined frequency? |
| **Location** | |
| 7 | Is the artifact stored in the project library in the location specified in the Software Development Plan? |
| **Configuration Management** | |
| 8 | If required, has the artifact been placed under configuration management? |

The metrics for architecture artifact is described below as shown in Table 6.

**Table 6.** Software Architecture Document; Artifact Specific Checklist

| **Software Architecture Document Specific Checklist Items** | |
|---|---|
| 1 | Does the Introduction provide an overview of the entire document? |
| 2 | Is the Purpose defined? |
| 3 | Is the Scope defined? |
| 4 | Are the definitions of terms, acronyms, and abbreviations defined? Note: This information may be provided by reference to the project's Glossary. |
| 5 | Are References been defined? |
| 6 | Is the organization of the Software Architecture Document been defined in the overview? |
| 7 | Is the Architecture of the system described in the Architectural Representation? |
| 8 | Are relevant views (i.e. Use Case, Logical, Process, Deployment, and Implementation) described in the architectural representation? |
| 9 | Have Architectural goals been identified? |
| 10 | Have Architectural constraints been identified? |
| 11 | Have architecturally relevant use cases been described in the Use Case View? |
| 12 | Have main use case been detailed with respective Use Case Realizations? |
| 13 | Have architecturally significant parts of the Design Model been described in the Logical View? |
| 14 | Is the Design Model decomposed in terms of the package hierarchy and layering in the Logical View overview? |
| 15 | Have architecturally significant Design Packages been detailed? |
| 16 | Have system threads and processes been described in the Process View? |
| 17 | Does the Software Architecture Document provide a view of the Deployment Model in the Deployment View section? |
| 18 | Does the Software Architecture Document describe the overall structure of the Implementation Model in the Implementation View section? |
| 19 | Is an overview of layering provided for the Implementation View? |
| 20 | Has the implementation layer been described in the layers section of the Implementation View? |
| 21 | Has a view of the Data Model been detailed in the Software Architecture Document? (optional) |
| 22 | Have dimensioning characteristics been described? |
| 23 | Have performance constraints been described? |
| 24 | Is the Software Architecture contributing to all capabilities of the system described (i.e. extensibility, reliability, portability)? |

## 7. METRICS FOR DESIGN MODEL

This artifact is an object model that explains the realization of use cases, and serves as an abstraction of the implementation model and the software program code. The design model is used as essential input to activities in implementation and test. It is a comprehensive and composite artifact encompassing all design classes, subsystems, packages, subsystems and

collaborations. The metrics for Design Model is described below as shown in Table 7. This is artifact specific metrics whereas for general artifact checklist we can refer to Table 5. This should be noted that the general artifact checklist as shown in Table 5 will be applicable to all the artifacts in the unified software development process. Design model and other models are constructed making use of the unified modeling language that is the analysis and design language in unified software development process. Again, we are making use of the checklists based approach as per CMMI [West,2004].

**Table 7.** Design Model Specific Checklist Items

| **Design Model Specific Checklist Items** | |
|---|---|
| 1 | Does the Design Model have a textual introduction? |
| 2 | Have Design Packages been described? |
| 3 | Do the Design Packages have brief descriptions? |
| 4 | Have the classes contained in the Design Package been defined? |
| 5 | Have the relationships inside the package been defined? |
| 6 | Have Design Packages contained inside other Design Packages been defined? |
| 7 | Have import dependencies with other packages been documented? |
| 8 | Have Design Subsystems been defined? |
| 9 | Do Design Subsystems include brief descriptions? |
| 10 | Are all realized interfaces clearly described? |
| 11 | Are all elements contained in the Subsystem defined? |
| 12 | Are dependencies with other design element documented? |
| 13 | Have Design Classes been defined? |
| 14 | Do relevant Design Classes include brief descriptions? |
| 15 | Have class responsibilities been defined? |
| 16 | Have the relationships of the Design Classes been defined? |
| 17 | Have operations of Design Classes been defined? |
| 18 | Have attributes of Design Classes been defined? |
| 19 | Are requirements associated with Design Classes referenced? |
| 20 | Have Interfaces been defined? |
| 21 | Do the Interfaces include brief descriptions? |
| 22 | Have the Interface operations been described? |
| 23 | Have relationships among Design Elements been defined? |
| 24 | Have design level Use Case Realizations been defined? |
| 25 | Has a textual "Flow of Events" been described for each use case realization? |
| 26 | Has an Interaction Diagram been defined for each Use Case Realization? |
| 27 | Has a Class Diagram been defined for each Use Case Realization? |
| 28 | Have the requirements associated with each Use Case Realization been described? |

## 8. QUANTIFICATION OF CHECKLIST ITEMS & DECISION MAKING

In order to quantify the metrics it is necessary to evaluate each checklist item and award a quantified value based on some scale. We have two process checklists that are general process checklist and process specific checklist. On the artifact side we have general artifact checklist and artifact specific checklist. We know that we have to quantify the attributes of evidence, FI/PI/NI/NA, issue # (category of problem), and comments. After setting the values based on the scale described below we can measure each of our checklist items in all the metrics for process

and artifacts. The Table 8 is prepared in such a way that based on the evaluation of the checklist item we can award the weight to each checklist item.

**Table 8.** Evaluation Scale

| **Evidence Values** | **FI/PI/NI/NA Values** | **Issue # Values** | **Comments Values** |
|---|---|---|---|
| Evidence in the form of template, artifact or guideline | As per Table 1. | Problem ID/Category | Comments by team |
| Very Strong-4 | FI-4 | Not Severe-4 | Strongly Recommended -4 |
| Strong-3 | PI-3 | Not Much Severe-3 | Recommended-2 |
| Sufficient-2 | NI-2 | Severe-2 | Recommended with Reservations-3 |
| Poor-1 | NA-NIL | Very Severe-1 | Not Recommended-1 |

We see that all the values are of the range from 1 to 4. We understand that score of 4 is for the best and score of 1 is the poorest indicator. Let us take an example of Software Architecture Document, and set the scale as follows for the artifact specific checklist as per Table 9. For the sake of convenience we are taking only five checklist items from Table 7.

**Table 9.** Quantifying Checklist Items

| **Software Architecture Document Specific Checklist Items** | | **Evidence Values** | **FI/PI/NI/NA Values** | **Issue # Values** | **Comments Values** |
|---|---|---|---|---|---|
| 1 | Does the introduction provide an overview of the entire document? | 4 | 4 | 2 | 4 |
| 2 | Is the purpose defined? | 3 | 4 | 1 | 4 |
| 3 | Is the scope defined? | 4 | 3 | 4 | 3 |
| 4 | Are the definitions of terms, acronyms, and abbreviations defined? Note: This information may be provided by reference to the project's Glossary. | 3 | 4 | 4 | 2 |
| 5 | Are references been defined? | 4 | 3 | 2 | 2 |
| Total | | 18 | 18 | 13 | 15 |
| Grand Total Out of 80 | | 64 | | | |

So there are five checklist items and four attributes for each checklist items. We know that number values that can be awarded to a particular cell are four and the minimum that can be awarded to is one. We have maximum of eighty points of evaluation. In the example we sum up each column and get the values. Finally a grand total is calculated as shown in the last row of the table. This number is the key to evaluation and we can make a decision that how much quality oriented the artifact is. We see that the Software Architecture Document could score 64 out of 80. We can conclude that it is 80 percent quality oriented.

## 9. SUMMARY

In this paper we investigated and understood the unified process workflow metrics from A&D perspective. We gave emphasis on major artifacts involved in these disciplines. There are particular roles to perform the activities. All these activities are streamed up in a workflow. When these activities are performed we need some artifacts as inputs. After the activities are done we receive some output artifacts. We have developed metrics for the major artifacts of these disciplines workflow. We have engineered up some metrics pertaining to the inputs and outputs. The metrics that are developed are for A&D Process Metrics, Analysis Model Artifact, Design Model Artifact, Software Architecture Document Artifact, Deployment Model Artifact and Data Model Artifact. Also we saw how to quantify the artifact checklist items and make a decision about the quality for different attributes.

## REFERENCES


[1] Ahern D A, Armstrong J, Clouse A, Ferguson J R, Hayes W, Nidiffer K E, "CMMI® SCAMPI Distilled Appraisals for Process Improvement," Addison-Wesley Professional, 2005.

[2] Burwick Diane, "How To Implement the CMMI - Real Process Improvement Using Proven Solutions," BPS Publishing, 2008.

[3] Haynes S R, Skattebo A L, Singel J A, Cohen, M A, Himelright J L, "Collaborative architecture design and evaluation," In Proceedings of the 6th Conference on Designing Interactive Systems, University Park, PA, USA, June 2006, pp. 219 – 228.

[4] Hitz M , Montazeri B, "Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective," IEEE Transactions on Software Engineering, Vol. 22, No. 4, April 1996, pp. 267 – 271.

[5] Jacobson I, Booch G, Rumbaugh J, "The Unified Software Development Process," Addison-Wesley Professional, 1999.

[6] Kannengiesser U, Zhu L, "An Ontologically-Based Evaluation of Software Design Methods," The Knowledge Engineering Review , Vol. 24, No. 1, Cambridge University Press, 2009, pp. 41-58

[7] Kazman R, Bass L, Abowd G, Webb M, "SAAM: A Method for Analyzing the Properties of Software Architectures," In Proceedings of 16th International Conference on Software Engineering, May 1994, Sorrento, Italy, pp. 81 – 90.

[8] Kazman R, Klein M, Barbacci M, Longstaff T, Lipson H, Carriere J, "The Architecture Tradeoff Analysis Method," In Proceedings of Fourth IEEE Conference on Engineering of Complex Computer Systems, Monterey, CA, USA, August 1998, pp. 68 – 78.

[9] Kruchten P B, "The Rational Unified Process: An Introduction," Addison-Wesley Professional, 2003.

[10] Langer M, "Analysis and Design of Information Systems," Springer, 2007.

[11] Mårtensson F, "Software Architecture Quality Evaluation Approaches in an Industrial Context," Licentiate Dissertation Series No. 2006:03, School of Engineering Department Of Systems And Software Engineering, Blekinge Institute Of Technology, Sweden, 2006

[12] Rational Unified Process®, Base Plug-in, Version 7.0.1, Based on: Base Concepts Plug-in Version: 1.0.1, © Copyright IBM Corp. 2010.

[13] Sharma M, Chandwani M, "Software Metrics and Object Oriented Business Modeling," In CSI-2006, 41st Annual National Convention of Computer Society of India, Kolkata, 23-24 November 2006, pp. 243-247.

[14] Sharma M, Chandwani M, "Quality in Business Modeling using UML," In Proceedings of 1st International Conference on Digital Information Management IEEE-ICDIM, Bangalore, 3-5 December 2006, pp. 294-299.

[15] Sharma M, Chandwani M, "Maturing capability in Unified Paradigm," In Proceedings of the International Conference on Advances in Computing, Communication and Control ACM SIGART, Mumbai, January 23-24, 2009, pp. 737-746.

[16] Uttangi R V, Rizwan S A, "Fast Track to CMMI Implementation: Integrating the CMMI and RUP Process Frameworks," IBM DeveloperWorks Rational Technical Library, 2007

[17] Wang M H, Lee C S, "An Intelligent PPQA Web Services for CMMI Assessment," In Eighth International Conference on Intelligent Systems Design and Applications, Vol.1, Kaohsiung, Taiwan, November 2008, pp. 229 – 234.

[18] West Michael, "Real Process Improvement Using the CMMI," Auerbach Publications, 2004.