CMMI

Capability Maturity Model (CMM):– To determine an organisation's current state of process maturity, the SEI uses an assessment that results in a five point grading scheme–The grading scheme determines compliance with aCMM that defines key activities required at different levels or process maturity.*CMMs contain the essential elements of effective processes for one or more bodies of knowledge.*•The CMM provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels-Level 1:Initial–Level 2:Repeatable–Level 3:Define–Level 4:Managed–Level 5: Optimizing. Level1-2:Disciplined process level2-3:Standard consistent process level3-4:predictable process level4-5:continously improving process. Used for•Software process improvement – An organization plans, develops, and implements changes to its software process.•Software process assessment:determine the state of an organization's current software process, determines the high-priority software process-related issues facing an organization•Software capability evaluation:identifies contractors who are qualified to perform the software work or monitors the state of the software process used on an existing software effort.• The components of the CMM-•Maturity levels–well-defined evolutionary plateau toward achieving a mature SW process.•Process capability–the range of expected results that can be achieved by following a software process-provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes•Key process areas–•Goals–•Common features–•Key practices(slide 15 16 17 insert pics)

Capability Maturity Model Integration (CMMI):- mission was to combine the following three source models into a single improvement framework for use by organizations pursing enterprise-wise process improvement:–Capability Maturity Model for Software (SW-CMM) v2.0 draft C–Electronic Industries Alliance Interim Standard (EIA/IS) 731 (SE-CM) – The Systems Eng CM–Integrated Product Development Capability Maturity Model (IPD-CMM) v0.98. CMMI support two different representations:–Staged–Continuous. Purpose:•A process is a leverage point for an organization's sustained improvement •The purpose of CMM Integration is to provide guidance for improving your organization's processes and your ability to manage the development, acquisition, and maintenance of products or services•CMMI places proven approaches into a structure that helps your organization appraise its organizational maturity or process area capability, establish priorities for improvement, and implement these improvements•CMMI places proven approaches into a structure that helps your organization appraise its organizational maturity or process area capability, establish priorities for improvement, and implement these improvements.

CMMI Framework: •The CMMI Framework addresses the following materials:–Model –Appraisal–Training •The model portion of the CMMI Framework is composed of the following:–CMMI Model Foundation–Shared material–Constellation-specific material:

•Acquisition(CMM-ACQ):provides guidance to enable informed and decisive acquisition leadership•Development(CMMI-DEV): provides guidance for measuring,monitoring and managing development processes•Services(CMMI-SVC):provides guidance for those providing services within organisations and to external customers.**CMMI for services purpose and sponsors:** extend the CMMI framework to cover the provision of services•The "CMMI for Services" project is sponsored by the following organizations:–CMMI Steering Group–Office of the Secretary of Defense for Acquisition, Technology, and Logistics–National Defense Industrial Association, Systems Engineering Division–Software Engineering Institute–Northrop Grumman.**Services:**A service is a product that is intangible and non-storable.**CMMI for Acquisition:**•A "CMMI for Acquisition" model will improve consistency and payoff for acquisition organizations because it will include acquisition practices that are useful, but not covered in the CMMI for Development models.

• build upon the initial work published by General Motors Corporation.**Purpose and Sponsor:** –CMMI Steering Group – Office of the Secretary of Defense for Acquisition,Technology,and Logistics–National Defense Industrial Association,Systems Engineering Division – General Motors–SoftwareEngineeringInstitute – NASA – Department of Homeland Security.**Acquisition:**• Acquisition, as defined in the CMMI models, is the process of obtaining products (goods and services) through contract.• Some call these processes "procurement" or "outsourcing."**Selecting a CMMI model:**• You must select a representation, either continuous or staged.• You must determine the bodies of knowledge you want to include in the model your organization will use– System Engineering– Software Engineering– Integrated Product and Process Development– Supplier Sourcing.**Representations:** two different ways:**(1)** as a continuous model:– Allow you to select the order of improvement that best meets the organization's objectives –Enable comparisons across and among organizations on a process area by process area basis or by comparing results through the use of equivalent staging

• Provide an easy migration from EIA/IS 731 to CMMI •Afford an easy comparison of process improvement to ISO/IEC 15504 **(2)** staged model:– Provide a proven sequence of improvements, beginning with basic management practices and progressing through a predefined and proven path of successive levels, each serving as a foundation for the next –Permit comparisons across and among organizations by the use of maturity levels• Provide an easy migration from the SE-CMM to CMMI• Provide a single rating that summarizes appraisal results and allows comparisons among organizations. **Integration Models:** 1.System Engineering:•covers the development of total systems, which may or may not include software•Systems engineers focus on transforming customer needs, expectations, and constraints into product solutions and supporting these product solutions throughout the life of the product.2.Software Engineering: covers the development of software systems Systems engineers focus on transforming customer needs, expectations, and constraints into product solutions and supporting these product solutions throughout the life of the product3.Integrated Product and Process Development (IPPD):•IPPD is a systematic approach that achieves a timely collaboration of relevant stakeholders throughout the life of the product to better satisfy customer needs, expectations, and requirements. 4.Supplier Sourcing **CMMI and Six Sigma:**•Six Sigma is an approach to business improvement that includes an improvement framework, performance measurement and a toolkit.**Process Area and Model Component:** 1.Required components: describe what an organization must achieve to satisfy a process area–Specific goals–Genericgoals• This achievement must be visibly implemented in an organization's processes– Goal satisfaction is used in appraisals as the basis for deciding whether a process area has been achieved and satisfied. 2. 2.Expected components:•describe what an organization may implement to achieve a required component •Expected components guide those who implement improvements or perform appraisals•Expected components include the–Specific practices–Generic practices

3.Informative components:•provide details that help organizations get started in thinking about how to approach the required and expected components.•Informative model components–Sub-practices, typical work products, amplifications (information relevant to a particular discipline), generic practice elaborations, goal and practice titles, goal and practice notes, references. **CMMI Models:** 1.Understanding Levels 2.Capability Levels 3.Maturity Levels. **Level:**•used to describe the evolutionary path recommended for an organization that wants to improve the processes it uses to develop and maintain its products and services.•CMMI supports two improvement paths•The first path enables organizations to incrementally improve processes corresponding to an individual process area (or process areas) selected by the organization.•The second path enables organizations to improve a set of related processes by incrementally addressing successive sets of process areas.•Two representations and levels–Continuous representation•Capability level –Staged representation•Maturity level •Levels characterize improvement from an ill- defined state to a state that uses quantitative information to determine and manage improvements that are needed to meet an organization's business objectives.**Continuous CMMI Meta-Model**: 1.Process Area 2.Capability level **Capability Levels:** •A capability level consists of related SPs and GPs for a process area that can improve the organization's processes associated with that process area.•Capability levels represent the process improvement attained in an organization **•Level 0:Incomplete**–An "incomplete process" is a process that is either not performed or partially performed–One or more of the specific goals of the process area are not satisfied and no generic goals exist for this level since there is no reason to institutionalise a partially performed process.•Level 1: Performed–A performed process is a process that accomplishes the needed work to produce work products; the specific goals of the process area are satisfied**•Level 2: Managed**– A managed process is a performed process (CL 1) that is planned and executed in accordance with policy; employs skilled people who have adequate resources to produce controlled outputs; involves relevant stakeholders; is monitored, controlled, and reviewed; and is evaluated for adherence to its process description.**•Level 3: Defined**:– A defined process is a managed process (CL 2) that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines, has a maintained process description; and contributes process related experiences to the organizational process assets.• **Level3:Defined(continued)**– A critical distinction between capability levels 2 and 3 is the scope of standards, process descriptions, and procedures.•At capability level 2, the standards, process descriptions, and procedures can be quite different in each specific instance of the process (e.g., on a particular project).•At capability level 3, the standards, process descriptions, and procedures for a project are more consistent, except for the differences allowed by the tailoring guidelines.• Another critical distinction is that at capability level 3 processes are typically described more rigorously than at capability level 2.**•High maturity levels:** the use of statistical and other quantitative techniques to improve organisational and project processes to better achieve business objectives.• **High maturity levels(continued):**it can continue its improvement journey by addressing high maturity process areas: Organizational Process Performance, Quantitative Project Management, Causal Analysis and Resolution, and Organizational Performance Management–After the organization attains CL 3 in the OPP and QPM process areas, the organization can continue its improvement path by selecting the CAR and OPM process areas. **Staged CMMI Meta-Model:** • The primary difference is that the staged model defines five maturity levels (ML1 – ML5) (rather than four capability levels (CL0 – CL3)).defined evolutionary plateau for organizational process improvement.• The maturity levels are measured by the achievement of the SGs and GGs associated with each predefined set of process areas. **Maturity Levels:** • Level 1: Initial:– processes are usually ad hoc and chaotic.–ML 1 organizations often produce products and services that work; however, they frequently exceed their budgets and do not meet their schedules.• Characterized by a tendency to over -commit, abandon their processes in a time of crisis, and be unable to repeat their successes.• Level 2: Managed: the projects have ensured that processes are planned and executed in accordance with policy• Level 2: Managed (continued):projects are performed and managed according to their documented plans.the status of the work products are visible to management at defined points. • Level3:Defined:processes are well characterised and understood, and are described in standards, procedures, tools, and methods. •Level4: Quantitatively Managed:the organization and projects establish quantitative objectives for quality and process performance – Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers.• Level 5: Optimizing:an organization continually improves its processes based on a quantitative understanding of its business objectives and performance needs.• Level 5: Optimizing (continued):– A critical distinction between maturity levels 4 and 5 is the focus on managing and improving organizational performance.– At maturity level 4, the organization and projects focus on understanding and controlling performance at the subprocess level and using the results to manage projects– At maturity level 5, the organization is concerned with overall organizational performance using data collected from multiple projects. Analysis of the data identifies shortfalls or gaps in performance. These gaps are used to drive organizational process improvement that generates measurable improvement in performance. **Process Area:**A process area is a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making significant improvement in that area. Represented in two ways:1. Continuous 2. Staged **Capability Level Profile vs. Maturity Level Rating:** • If the continuous representation is used for an appraisal, the rating is a "capability level profile."• If the staged representation is used for an appraisal, the rating is a "maturity level rating".**Equivalent Staging:** Equivalent staging is a way to compare results from using the continuous representation to results from using the staged representation•A capability level profile is a list of process areas and the corresponding capability level achieved for each.•This profile enables an organization to track its capability level by process area.•The profile is called an"achievement profile" when it represents the organization's actual progress for each process area.•Alternatively, the profile is called a "target profile" when it represents the organization's planned process improvement objectives. The following rules summarize equivalent staging:–To achieve maturity level 2, all process areas assigned to maturity level 2 must achieve capability level 2 or 3–To achieve maturity level 3, all process areas assigned to maturity levels 2 and 3 must achieve capability level 3–To achieve maturity level 4, all process areas assigned to maturity levels 2, 3, and 4 must achieve capability level 3.–To achieve maturity level 5, all process areas must achieve capability level 3.**CMMI Process Areas:**1.Process Management:process areas contain the cross-project activities related to defining, planning, deploying, implementing, monitoring, controlling, appraising, measuring, and improving processes. The Process Management PAs:– Organizational Process Definition (OPD)– Organizational Process Focus (OPF)– Organizational Performance Management (OPM) – Organizational Process Performance (OPP)– Organizational Training (OT) 2.Project Management:The Project Management PAs:– Integrated Project Management (IPM)– Project Monitoring and Control (PMC)– Project Planning (PP)– Quantitative Project Management (QPM) – Requirements Management (REQM)– Risk Management (RSKM)– Supplier Agreement Management (SAM) 3.Engineering:cover the development and maintenance activities that are shared across engineering discipline.The Engineering PAs:– Product Integration (PI)– Requirement Development (RD) – Technical Solution (TS)– Validation (VAL)– Verification (VER)

4.Support: The Support PAs– Causal Analysis and Resolution (CAR)– Configuration Management (CM)– Decision Analysis and Resolution (DAR)– Measurement and Analysis (MA)– Process and Product Quality Assurance (PPQA) **Basic Support Process Areas:**117-120 **CMMI Models: Interpreting:**•best practice used improve their processes for developing, acquiring, and maintaining products and services.•map your improving processes to CMMI process areas•This mapping enables you to track your organization's level of conformance to the CMMI model you are using and to easily identify opportunities for improvement. **Interpreting CMMI When Using Agile Approaches:** •CMMI does not endorse any particular approach to development, little information•for agile methods, notes have been added to selected process area. **Using CMMI Appraisals:**find value in measuring their progress by conducting an appraisal and earning a maturity level rating or a capability level achievement profile.• Appraisal teams use CMMI to guide their identification and prioritization of findings. **Appraisal Requirements for CMMI:**describes the requirements for several types of appraisals. **SCAMPI Appraisal Methods:**accepted method used for conducting ARC Class A appraisals using CMMI models. **Appraisal Consideration**: • Choices that affect a CMMI-based appraisal include the following:

–CMMI model–Appraisal scope, including the organizational unit to be appraised, the CMMI process areas to be investigated, and the maturity level or capability levels to be appraised–Appraisal method–Appraisal team leader and team members– Appraisal participants selected from the appraisal entities to be interviewed–Appraisal outputs (e.g., ratings, instantiation-specific findings)–Appraisal constraints (e.g., time spent on site) **Considerations:** •appraisal principles for CMMI:–Senior management sponsorship–A focus on the organization's business objectives– Confidentiality for interviewees–Use of a documented appraisal method–Use of a process reference model (e.g., a CMMI model)–A collaborative team approach–A focus on actions for process improvement **Appraisals and Benchmarking:** The International Organization for Standardization and International Electro-technical Commission is an emerging standard for SW process– SPICE (Software Process Improvement and Capability dEtermination) **Process Areas Goals and Practices**: –Specific Goals establish the characteristics that must exist if the activities implied by a process area are to be effective–Specific practices refine a goal into a set of process-related activities.•The CMMI also defines a set of three generic goals (GG) and related practices (GP) for each process area.•to achieve a particular capability level, the generic goal for that level and the generic practices that correspond to that goal must be achieved. **Process Institutionalisation:** the process is ingrained in the way the work is performed.–"That's the way we do things around here."•The organization builds an infrastructure that contains effective, usable, and consistently applied processes.–The organizational culture conveys the process, and management nurtures the culture.–Culture is conveyed through role models and recognition.– Institutionalized processes endure after the people who originally defined them have gone. **Generic Goals and Institutionalization:**GG1:Achieve specific goals : Performed process, GG2:Institutionalise a managed process:Managed process, GG3:Institutionalise a Defined process:Defined Process.

Mat. Level	Focus	Process Areas
5. Optimizing	Continuous process improvement	Organizational Performance Management (OPM) Causal Analysis and Resolution (CAR)
4. Quantitatively Managed	Quantitative management	Organizational Process Performance (OPP) Quantitative Project Management (QPM)
3. Defined	Process standardization	Requirements Development (RD), Technical Solution (TS), Product Integration (PI), Verification (VER), Validation (VAL), Organization Process Focus (OPF), Organizational Process Definition (OPD), Organizational Training (OT), Integrated Project Management (IPM) Risk Management (RSKM), Decision Analysis and Resolution (DAR
2. Managed	Basic project management	Requirement Management (REQM), Project Planning (PP), Project Monitoring and Control (PMC), Supplier Agreement Management (SAM), Measurement and Analysis (MA), Process and Product Qual Assurance (PPQA), Configuration Management (CM)

Process Areas Presented in the Staged Representation: Maturity Level 2 • Requirements Management (REQM)• Project Planning (PP)• Project Monitoring and Control (PMC) • Supplier Agreement Management (SAM)• Measurement and Analysis (MA)• Process and Product Quality Assurance (PPQA) • Configuration Management (CM) **Requirement Management (REQM)** • An Project Management PA at ML 2 • The purpose of Requirements Management (REQM) is to manage requirements of the project's products and product components and to ensure alignment between those requirements and the project's plans and work products.• Specific Goals and Practices• SG 1 Manage Requirements – SP 1.1 Understand Requirements – SP 1.2 Obtain Commitment to Requirements – SP 1.3 Manage Requirements Changes – SP 1.4 Maintain Bidirectional Traceability of Requirements – SP 1.5 Ensure Alignment Between Project Work and Requirements **Project Planning (PP):**to establish and maintain plans that define project activities.• The Project Planning process area involves: – Developing the project plan– Interacting with stakeholders appropriately– Getting commitment to the plan– Maintaining the plan.• Specific Goals and Practices• SG 1 Establish Estimates:– SP 1.1 Estimate the Scope of the Project• Establish a top-level work breakdown structure (WBS) to estimate the scope of the project. • Example Work Products– Task description, work package descriptions, WBS• Sub-practices 1. Develop a WBS.2. Define the work packages in sufficient detail so that estimates of project tasks, responsibilities, and schedule can be specified.3. Identify product or product components to be externally acquired. 4. Identify work products to be reused.– SP 1.2 Establish Estimates of Work Product and Task Attributes – SP 1.3 Define Project Lifecycle Phases – SP 1.4 Estimate of Effort and Cost

• SG 2 Develop a Project Plan:– SP2.x...• SG 3 Obtain Commitment to the Plan:– SP3.x…

•Generic Practices by Goal– GG 1 Achieve Specific Goals • GP 1.1 Perform Specific Practices – GG 2 Institutionalize a Managed Process • GP 2.1 Establish an Organisational Policy • GP 2.2 Plan the Process •– GG 3 Institutionalize a Defined Process **Project Monitoring and Control (PMC):**PA at ML 2 • The purpose of Project Monitoring and Control (PMC) is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan.

• Project plan • Corrective Actions– Re-planning, establishing new agreements, including additional mitigation activities in the plan. **PMC goals and practices**:• SG 1 Monitor the Project Against the Plan – SP 1.1 Monitor Project Planning Parameters – SP 1.2 Monitor Commitments– SP 1.3 Monitor Project Risks– SP 1.4 Monitor Data Management– SP 1.5 Monitor Stakeholders Involvement – SP 1.6 Conduct Progress Reviews– SP 1.7 Conduct Milestone Reviews• SG 2 Manage Corrective Action to Closure – SP 2.1 Analyze Issues– SP 2.2 Take Corrective Action– SP 2.3 Manage Corrective Actions **Supplier Agreement Management (SAM):** • The purpose of Supplier Agreement Management (SAM) is to manage the acquisition of products and services from suppliers.•The SAM process area involves the following activities: – Determining the type of acquisition– Selecting suppliers– Establishing and maintaining agreements with suppliers– Executing supplier agreements– Accepting delivery of acquired products– Ensuring successful transition of acquired products

Measurement and Analysis (MA):• The purpose of Measurement and Analysis (MA) is to develop and sustain a measurement capability used to support management information needs.• The Measurement and Analysis process are involves the following:– Specifying objectives of measurement and analysis– Specifying measures, analysis techniques, and mechanism for data collection, data storage, reporting and feedback– Implementing the analysis techniques and mechanisms for data collection, data reporting and feedback– Providing objective results that can be used in making informed decisions and taking appropriate corrective action. **Process and Product Quality Assurance (PPQA):**provide staff and management with objective insight into processes and associated work products.• The PPQA process area involves:– Objectively evaluating performed processes and work products against the applicable process descriptions, standards, and procedures– Identifying and documenting noncompliance issues– Providing feedback to project staff and managers on the results of QA activities– Ensuring that noncompliance issues are addressed. **Configuration Management (CM):** to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits.process involves– Identifying the configuration of selected work products that compose the baselines at given points in time–Controlling changes to configuration items–Building or providing specifications to build work products from the configuration management system–Maintaining the integrity of baselines–Providing accurate status and current configuration data to developers, end users, and customers. Process Areas Presented in the Staged Representation: Maturity Level 3• Requirements Development (RD)

•Technical Solution (TS)• Product Integration (PI)• Verification (VER)• Validation (VAL)• Organizational Process Focus (OPF)• Organizational Process Definition (OPD)• Organizational Training (OT)• Integrated Project Management (IPM)• Risk Management (RSKM)• Decision Analysis and Resolution (DAR) **Process Areas Presented in the Staged Representation: Maturity Level 3:Requirements Development (RD):**to elicit, analyze, and establish customer, product, and product component requirements.•RD describes three types of requirements:–Customer requirements –Product requirements–Product component requirements. **Technical Solution (TS):**to select, design, and implement solutions to requirements. • This process area focuses on the following:

– Evaluating and selecting solutions (referred to as "design approaches," "design concepts," or "preliminary designs") – Developing detailed designs for the selected solutions (detailed in the context of containing all the information needed to manufacture, code) – Implementing the designs as a product or product component **Product Integration (PI):**to assemble the product from the product components, ensure that the product, as integrated, behaves properly (i.e., possesses the required functionality and quality attributes), and deliver the product. **Verification (VER):** ensure that selected work products meet their specified requirements.•The VER PA involves the following: – Verification preparation– Verification performance– Identification of corrective action•Verification includes verification of the product and intermediate work products against all selected requirements, including customer, product, and product components requirements.**Validation (VAL):**demonstrate that a product or product component fulfills its intended use when placed in its intended environment **Organizational Process Focus (OPF):**to plan, implement, and deploy organizational process improvements based on a thorough understanding of the current strengths and weakness of the organization's processes and process assets **Organizational Process Definition (OPD):**to establish and maintain a usable set of organizational process assets and work environment standards, and rules and guidelines for teams. **Organizational Training (OT):**to develop the skills and knowledge of people so they can perform their roles effectively and efficiently.– Identifying the training needed by the organization– Obtaining and providing training to address those needs– Establishing and maintaining training capability– Establishing and maintaining training records– Assessing training effectiveness **Integrated Project Management (IPM):**establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard processes.**Risk Management (RSKM):**identify potential problems before they occur so that risk- handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives.• RSKM can be divided into three parts:

–Defining a risk management strategy– Identifying and analyzing risks–Handling identified risks, including the implementation of risk mitigation plans (and contingency plan) when needed **Decision Analysis and Resolution (DAR):**to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria. **Process Areas Presented in the Staged Representation: Maturity Level 4: Organizational Process Performance (OPP):**to establish and maintain a quantitative understanding of the performance of selected processes in the organization's set of standard processes in support of achieving quality and process performance objectives, and to provide process performance data, baselines, and models to quantitatively manage the organization's projects. **Quantitative Project Management (QPM):**to quantitatively manage the project to achieve the project's established quality and process performance objectives.following activities:– Establishing and maintaining the project's quality and process performance objectives – Composing a defined process for the project to help to achieve the project's quality and process performance objectives – Selecting subprocesses and attributes critical to understanding performance and that help to achieve the project's quality and process performance objectives – Selecting measures and analytic techniques to be used in quantitative management – Monitoring the performance of selected subprocesses

using statistical and other quantitative techniques– Managing the project using statistical and other quantitative techniques to determine whether or not the project's objectives for quality and process performance are being satisfied– Performing root cause analysis of selected issues to address deficiencies in achieving the project's quality and process performance objectives **Process Areas Presented in the Staged Representation Maturity: Level 5:Organizational Performance Management (OPM):**to proactively manage the organization's performance to meet its business objectives.• Business objectives that this process area might address include the following:– Improved product quality(e.g.,functionality,quality attributes)–Increased productivity–Increased process efficiency and effectiveness–Increased consistency in meeting budget and schedule –Decreased cycle time –Greater customer and end-user satisfaction–Shorter development or production time to change functionality,add new features, or adapt to new technologies–Improved performance of a supply chain involving multiple suppliers – Improved use of resources a cross the organization. **Causal Analysis and Resolution (CAR):**identify causes of selected outcomes and take action to improve process performance.• The CAR PA involves the following:–Identifying and analyzing causes of selected outcomes (e.g., defects and other problems).–Taking specific actions to complete the following: • Remove the causes and prevent the occurrence of those types of defects and problems in the future.• Proactively analyze data to identify potential problems and prevent them from occurring• Incorporate the causes of successes into the process to improve future process performance.

SCAMPI

Objectives: – Who want to learn CMM/CMMI– to understand and improve their capability to develop software effectively– to understand the key practices that are part of effective processes for developing or maintaining software, and to identify the key practices that are needed to achieve the next maturity level in the CMMI –to identify the risks of having a particular organization perform the work of a contract. Appraisals and Benchmarking: find value in benchmarking their progress (e.g., ascertaining maturity level scores, a capability level profile) in process improvement for both internal purposes and with external customers and suppliers. • Process appraisals focus on identifying improvement opportunities and understanding the organization's position relative to the selected model or standard.

•uses CMMI to guide their identification and prioritization of findings.• The appraisal principles for the CMMI Product Suite remain the same as those used in appraisals for other process-improvement models.• Those principles are:– Senior management sponsorship –A focus on the organization's business objectives–Confidentiality for interviewees–Use of a documented appraisal method–Use of a process reference model (CMMI) as a base–A collaborative team approach–A focus on actions for process improvement. • It is called the Standard CMMI Appraisal Method for Process Improvement (SCAMPISM). SCAMPI MDD:

•The Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) A is designed to provide benchmark quality ratings relative to Capability Maturity Model® Integration (CMMI) models. • describes the requirements, activities, and practices associated with each of the processes that compose the SCAMPI A method.– Precise Listing Of Required Practices, parameters, and variation limits, as well as optional practices and guidance for enacting the method, are covered. People who will enact an appraisal (appraisers) are expected to read the second part of the document (MDD). • MDD's Part II is divided into Process Definitions, which are in turn divided into Activity Descriptions.

•The processes (as defined) support a variety of orderings and enactments to facilitate a variety of usage modes for SCAMPI A.• The temporal flow, as well as the flow of inputs and outputs among the processes, is described in the Method Overview section. The Process Definitions are not intended to provide a start-to-finish view of SCAMPI A.• Instead, these sections provide detailed definitions of processes and activities that are invoked according to the appraisal plan created by the appraisal team leader. SCAMPI Appraisal Classes: includes A,B and C. SCAMPI A only result in rating • SCAMPI B provides options in model scope, but the characterization of practices is fixed to one scale and is performed on implemented practices. SCAMPI C provides a wide range of options, including characterization of planned approaches to process implementation according to a scale defined by the user.

- Characteristics of Appraisal Classes

Characteristic	Class A	Class B	Class C
Amount of objective evidence	High	Medium	Low
Ratings generated	Yes	No	No
Resource needs	High	Medium	Low
Team size	Large	Medium	Small

• best practices include: – CMM-Based Appraisal for Internal Process Improvement (CBA IPI) V1.1– Electronic Industries Alliance/Interim Standard (EIA/IS) 731.2 Appraisal Method– Software Capability Evaluation (SCESM) V3.0 Method Description– Software Development Capability Evaluation (SDCE) –FAA Appraisal Method (FAM)

Indicator Type	Description	Examples
Direct artifacts	The tangible outputs resulting directly from implementation of a specific or generic practice. An integral part of verifying practice implementation. May be explicitly stated or implied by the practice statement or associated informative material.	Typical work products listed in reference model practices Target products of an "Establish and Maintain" specific practice Documents, deliverable products, training materials, etc.
Indirect artifacts	Artifacts that are a consequence of performing a specific or generic practice or that substantiate its implementation, but which are not the purpose for which the practice is performed. This indicator type is especially useful when there may be doubts about whether the intent of the practice has been met (e.g., an artifact exists but there is no indication of where it came from, who worked to develop it, or how it is used).	Typical work products listed in reference model practices Meeting minutes, review results, status reports, presentations, etc. Performance measures
Affirmations	Oral or written statements confirming or supporting implementation (or lack of implementation) of a specific or generic practice. These statements are usually provided by the implementers of the practice and/or internal or external customers, but may also include other stakeholders (e.g., managers and suppliers).	Instruments Interviews Presentations, demonstrations, etc.

Determine Process Area Ratings: process area ratings are selected: using a continuous representation, assign a capability level to each process area within the scope of the appraisal, based on the highest level for which all specific goals and generic goals within the appraisal scope have been satisfied. If using a staged representation, rate the satisfaction of each process area within the scope of the appraisal, based on the satisfaction ratings assigned to all goals included in that process area. • If any of the goals are rated Not Rated and none of the other goals are rated Unsatisfied, then the process area is rated Not Rated.

• When a process area is determined to be outside of the organizational unit's scope of work, the process area is designated as "Not Applicable" and is Not Rated.• When an applicable process area is outside of the scope of the model used for the appraisal, the process area is designated as "Out of Scope" and is Not Rated.• If using a staged representation, rate the satisfaction of each PA.– PAs must be assigned rating values of Satisfied, Unsatisfied, Not Applicable, Out of Scope, or Not Rated.– A PA is rated Satisfied if and only if all of its SGs and GGs associated with a given maturity level and below are rated Satisfied.– If even one of the goals associated with a given maturity level in a PA is rated Unsatisfied, then the PA is rated Unsatisfied for that maturity level and above.• For an appraisal using a continuous representation, the following table defines the basis for CL ratings. • For an appraisal using a staged representation model, the "satisfied" rating for a PA may depend on the target ML for the appraisal. – Rating a PA "staged at ML 2" as "satisfied" would require a satisfied rating for GG2 in order to support a ML 2 outcome. Determine Process Area Profile: • The appraisal team may create a process area profile (called "capability level profile" or "achievement profile" in CMMI models) that graphically depicts the ratings assigned to each process area within the scope of the appraisal. A simple bar chart can be used for the display. Each process area is represented in a single bar along the horizontal axis, and the vertical axis represents the rating dimension.• Capability levels take only the values 0, 1, 2, or 3. Intermediate values are not defined for this appraisal outcome, and any embellishment of the Capability Profile with such values is outside the boundaries of SCAMPI A.• Satisfaction ratings, which may take on one of two values "Satisfied" or "Unsatisfied" for each process area, are used when the appraisal is using the staged representation. Determine Maturity Level: • When using the Staged Representation, the maturity level determined is the highest level at which all PAs contained within the maturity level, and within all lower maturity levels, are rated as Satisfied or Not Applicable.– The single exception to this rule is that generic goal 3 for applicable maturity level 2 PAs must also be rated Satisfied for a maturity level rating of 3 or higher to be determined.• When using the Continuous Representation, the appraisal reference model provides for equivalent staging, whereby a Capability Profile can be used to derive an equivalent Maturity Level rating.–A maturity level for a continuous representation is achieved if the Capability Profile is at or above the target profile for all PAs for that maturity level and all lower maturity levels in the equivalent staging, excepting those PAs that are designated as Not Applicable. Document Appraisal Result: • Appraisal team shall– document the final findings– document the rating outcome(s)– document the Appraisal Disclosure Statement (ADS) • The ADS is a summary statement describing the appraisal results that includes the conditions and constraints under which the appraisal was performed. • ADS contains information considered essential to adequately interpret the meaning of assigned maturity level or capability level ratings.• The ADS is prepared by the appraisal team leader and provided to the appraisal sponsor.

CH 12. Software Configuration Management (II)

Software Configuration Management: preserve the integrity of a software design throughout its useful life, it must be maintained under SCM control. **Programs:** long lives, changes made to correct errors, make enhancements → design updates req. [design is not maintained, these subsequent changes must be made with an incomplete knowledge of the design or the logic behind it] **Make design changes:** provide for their orderly implementation and test, control maintained over items → • Operational concept • Requirements • Specifications • Design documents • Source code • Object code • Test plans • Test cases, test configurations, and test results • Maintenance and development tools • User manuals • Maintenance manuals • Interface control documents. **Example Baseline Contents** – To control items and not to constrain early development excessively, design baselines are established at appropriate points. – While the items are relevant to the initial software development project, many of them are needed for the subsequent repair and enhancement effort as well.

Software Configuration Management Plan: establishing an SCM system → develop a Software Configuration Management Plan. **SCM Plan:** objectives, responsibilities & the approach, and methods to be used. Methods: Example SCMP Contents [Overview • SCM Organization • SCM Methods • SCM Procedures • SCM Implementation], SCMP Checklist [General • Organization • Identification • Control • Status Accounting • Audits • Program Phasing • Subcontractor/Vendor • Interface Control]

Software Configuration Management Questions:

1. **Requirements phase**: [– Where is the official requirement statement? – What changes have been made, and have they been approved? – How much change activity has there been, and what is the impact on design and/or development? – How many requirements changes have been made without changes in the contract scope?]
2. **Design phase:** [– Where is this particular requirement covered by the design? – What is the particular requirement that this design element satisfies? – What is the current approved specification for this interface? – Has the format for this data item been specified, where, by whom, and who is using it? – What is the design impact of this requirements change? – What tools and methods were used to develop this design?]
3. **Implementation phase:** [– How has this particular function been allocated to the various implementation areas, and why? – What is the design logic for this particular code element? – How is the design affected by this code change? – What is the implementation impact of this requirements change? – What compiler version was used to produce this code?]
4. **Testing phase:** [– Where are the tests that verify this functional requirement? – Where is the test data for use with these tests? – Where is the documentation that explains the results to be expected from these tests? – Has this test bucket been updated to reflect the functional content of the latest release or system build? – What tests have been run on this program version, and what were the results? – What test tools and support facilities were used for these tests?]

SCM Support Functions: To support SW activities, sophisticated set of SCM functions is req. --- used in varying degrees during requirements, design, implementation, test, and maintenance, all required to provide adequate support and effective control. **Functions:** A *protected baseline* for the operational concept, specifications, design, implementation, test, and tools • A *protected file* with a description of all changes and revisions • Means for each software engineer to read any *unlocked element* in the baseline • A *private workspace* where charged-out designs or implementations can be modified • *Templates* that assist in the preparation of new design, implementation, or test descriptions • A procedure for making approved *charge-outs* that permits SW engineers to obtain any available baseline element for their private workspace and *lock out the SCM copy* function to prevent anyone else from making simultaneous changes • A *charge-in procedure* • A procedure for making *approved deletions of defunct elements* • A way to collect, format, and produce *consolidated system documents* containing the key element descriptions for any given baseline • A way to check that all elements and relevant descriptions have *been carried over between baselines* • A *centralized data dictionary* containing the official record of all named items and their formats • A *where-used record* of every use of every interface and data item in the system

The requirements phase: ensure that the users' needs are properly understood before designing and implementing a system to meet them. • The requirements also provide part of the basis for system and acceptance testing • early design will likely be changed many times • requirements cannot be completely specified at the beginning of the project, they must be gradually evolved [initial requirements agreement is reached, a baseline is established as a design foundation, as a basis for change negotiations, and as a reference point for acceptance testing.]

Requirements Baselines and Change Control: • Each change is submitted on a design change request (DCR) form and reviewed, approved, and tracked with the same discipline as design and code changes. All requirements' changes are handled in this way, whether submitted by the customer, systems design, development, or test. Without such a discipline, baseline integrity for the entire project is exposed. • Essential to establish project naming conventions. • The Software Configuration Identification (SCI) system uniquely identifies every project development item. • The SCI definitions are kept under configuration control and expanded as more is learned about the product and its structure. • SCI Tree Structure • Design Change Request Form

External Specifications: • detailed specifications are developed for each program component. • While these specifications may all be produced at one time, generally some of the component specs are evolved in concert with development.

Design Control: • During design, the Change Control Board (CCB) controls the design changes, and SCM maintains complete records of every change and its rationale. • The Software Configuration Identification then connects each design element back to the specifications. • A good design requires a clear problem statement. • The design is the template against which the implementers produce the code; it must be clear, unambiguous, and absolutely current. • A permanent design reference is required for later repair or product extension. • A permanent record is needed of the design changes, why they were made, and how they were implemented. • To maintain control as the design is subdivided into modules, an SCM procedure is required to check the self-consistency of the modules and interface descriptions in each design baseline. • The design phase continues until the smallest structural elements (e.g., modules, packages) of the systems are defined and specified. • The detailed design phase then starts with these module specifications. • Example Module Specification • Additional SCM facilities needed for detailed design: – Template for module specifications and design skeletons, etc.

Implementation Phase: **Design-related capabilities:** – First, with a fully controlled design, implementation can start on a solid foundation. – As implementation proceeds, design problems will be found, and design changes will be needed. – When such changes are made, the design documentation is updated.

Operational data: Most large-scale software systems generally require large amounts of specialized data that must be specified, produced, tested, documented, and controlled. It is thus essential that SCM provisions be made to handle this data.

Test phase: • Once a newly developed program is put in a baseline, all further changes must be controlled. • The major concerns are problem incidents, temporary patches, permanent bug fixes, and enhancements. • Precise and complete records must be maintained of every test run, the data and test case used, the configurations on which they were run, and the results. • Implementation Change Control Procedure

SCM for Tools: need to maintain baseline control over the complete set of tools used to specify, design, implement, and test the software. – One approach is to maintain a standard tool set under configuration control and to establish a new tool baseline whenever any change is made. – Each module specification, design, and code record then include a notation of the tool baseline used in its production. – A summary of this data should also be retained with the module revision history

Configuration Accounting: • purpose - maintain a continuous record of the status of all baselined items. • This record is not only a useful management tool, but, if old records are routinely archived, it also provides valuable insurance against disaster. • The information required for SC status accounting: – The time at which each baseline was established – When each SW configuration item and change was included in the baseline – A description of each configuration item – The status of each SW-related engineering change – The description of each SW change – The status of each SW change – The documentation status for each baseline – The changes planned for each identified future baseline

Software Configuration Audit: • A software configuration audit should periodically be performed to ensure that the SCM practices and procedures are rigorously followed. • This audit can be performed directly by the SCM staff or by some independent assurance function. • It is generally advisable to have SCM conduct frequent self-audits with an independent party occasionally performing spot checks.

CH 13. Defining the Software Process

Defining the Software Process: SW development - exceedingly complex, many alternative ways to perform the various tasks. A defined process can help guide the SW professionals through these choices in an orderly way. • With an established process definition, they can better understand – what they should do, – what they can expect from their co-workers, and – what they are expected to provide in return. • Factors to be considered: – Since software projects have differences, their SE processes must have difference as well. – In the absence of a universal SE process, organizations and projects must define processes that meet their own unique needs. – The process used for a given project must consider the experience level of the members, current product status and the available tools and facilities.

Process Standards: • While the need for project-unique process definitions is clear, there are also compelling reasons for standardization: – Process standardization helps to reduce the problems of training, review, and tool support. – With standard methods, each project's experiences can contribute to overall process improvement. – Process standards provide the basis for process and quality measurements. – Since process definitions take time and effort to produce, it is impractical to produce new ones for each project. • The conflicting needs for customization and standardization can often be resolved by establishing a process architecture, which consists of a standard set of unit or kernel process steps with rules for describing and relating them. • Customization is then achieved through appropriate interconnections of these standard elements into tailored process models.

Definitions: • Software engineering – The disciplined application of engineering, scientific, and mathematical principles, methods, and tools to the economical production of quality software • Software process – The set of activities, methods, and practices that are used in the production and evolution of software • Software engineering process – The total set of SE activities needed to transform a user's requirements into software • Software process architecture – A framework within which project-specific software processes are defined • Software process model – One specific embodiment of a software process architecture

Levels of Software Process Models: Software process models can be defined at any of three levels. – The U, or Universal, process model provides a high-level overview. – The W, or Worldly, process model is the working level that is familiar to most programmers and managers. – The A, or Atomic, process model provides more detailed refinements.

• Waterfall Model is still the best known and most widely used overview framework for the software development process. The Waterfall Process Model – System requirements – Software requirements – Analysis – Program design – Coding – Testing • Spiral Model

U Process Models: they represent the general workflow and provide overview understanding, they are not easily decomposed into the progressively finer levels of detail that are needed to guide the work of the software professionals. * Views – Task-oriented view – Functional view – Behavioral view – Structural view – Conceptual view – Entity view

A Process Models: At the opposite extreme from U-level models, Atomic- (A-) level process models can be enormously detailed. • They are needed by anyone who attempts to automate a specific process activity or use a standardized method or procedure to guide execution of a task. • Precise data definition, algorithmic specification, information flows, and user procedures are essential at this level. • The amount of detail to be included in such models must be determined by their use. • Atomic process definitions are often embodied in process standards and conventions.

W Process Models • The Worldly (W) process level is of most direct interest to practicing SE. – It guides the sequence of their working tasks and defines task prerequisites and results. – When reduced to operational

form, these models generally look like procedure. – They specify who does what when. – Where appropriate, they reference the A level that specifies standard task definitions or tool usage. – For each task, W models define the anticipated results, the appropriate measures, and the key checkpoints.

Examples of Process Models: The three levels of process models can be viewed as embodied in – Policies at the U level – Procedures at the W level – Standards or tools at the A level • *U level*, policies establish a high-level framework and set of principles that guide the overall behavior of organizations. • They are particularly helpful in unanticipated circumstances where no precedents have been established. • Examples – All work will be subjected to an inspection before it is incorporated in a baseline. • *W level*, procedures are established to implement the policies. • This W-level process model refers to any available Atomic-level standards that define precisely how tasks are to be performed. • Examples – A procedure might define the points at which QA reviews are to be conducted and how the resulting issues are to be handled. • Atomic level standards then serve as the basis for directing the work and for the SQA review. – For example, a code inspection standard would specify what code is to be reviewed, when, the methods to be used, the reports to be produced, and the acceptable performance limits.

Prescriptive and Descriptive Uses of Models: Process models can be used either to describe what is done or to characterize what is supposed to be done. – In a descriptive case, models can provide useful information about the process and its behavior. – This book uses process models in a prescriptive sense. • The approach is to define how the process should be conducted and suggest where appropriate policies, procedures, and standards could help guide the work.

Software Process Architectures: Since most organizations have at least some policies, procedures, and standards, they are also generally following some intuitive U-, W-, and A-level models both prescriptively and descriptively. To be fully effective, the process models should be explicit and should relate to each other. An architectural framework is needed to define the basic elements, how they relate, and how they are decomposed into greater detail.

Critical Software Process Issues: The reason for defining the software process is to improve the way the work is done. • By thinking about the process in an orderly way, it is possible to anticipate problems and to devise ways to either prevent or to resolve them. • Some of the major software process issues concern – Quality – Product technology – Requirements instability – Complexity • Requirements instability – Unknown requirements – Unstable requirements – Misunderstood requirements

Preliminary Process Architecture: Organizations that face the issues of quality, product technology, requirements stability, and/or complexity need to define ways to address them. Process architecture permits them to represent and manipulate the process at the U level and then selectively to refine it to the W and A levels. • This, or course, needs an overall architectural framework and a set of definitions. • Basic Unit Cell– The basic elements of the process architecture – unit cells – Each cell is defined to accomplish a specified task and is uniquely identified. – Each cell also has required entry conditions specified for task initiation that include the inputs (one or more with their sources). – The task standards, procedures, methods, responsibilities, and required measurements are also defined. – The exit conditions define the results produced, their level of validation, and any other post-task conditions. – Cell feedback refers to any data provided to or received from other stages in the process. – Single-Cell Development Process – Shows a full development cycle in one cell – Shows what kind of information is required for every process cell • The U-Level Development Process– More refined U-level model of the development process. The development cycle is broken into the basic cells of the Waterfall model. • U-Level Development Process Cell Specifications

Standard Process Elements: When looked at from the highest, or U, level, software processes tend to look much the same. • This is because they are described in broad generic terms like design, implementation, or test. • When these activities are broken into more detail, however, significant differences show up. • With all variation at the W-level, however, many software activities can be relatively standardized across different projects. • It is thus possible to establish some basic process cells that can be interconnected in different ways to meet project-unique needs. • The detailed structure of these standard cells is then further defined by A-level models as needed.

Implementation Cells: In defining a standard set of software process cells, we start with some relatively detailed software tasks. – Basic Implementation Cell C_0– Quick Kernel K_0 – Inspection Operator

Unit Implementation Kernel: Implementation Cell C_1 • Unit Implementation Kernel K_1

Cell Specification: Once the general flow of a process is known, it is important to define each process cell. • ETVX paradigm – Entry – Task – Verification – Exit • ETX specification – A modified version – The explicit characterization of the Entry, Task, and Exit criteria for each process action • ETX Specification for the Implementation Cell C_1 • ETX Specifications for the Unit Implementation Kernel K_1 • These process specifications for each project should include explicit responsibilities for task performance and should refer to the applicable standards and procedures. – A refined process can provide software engineers guidance as well as setting the standards for management review and SQA audit.

Larger Process Models: • Once some basic process cells have been defined, it is possible to construct larger process models. • This is done by interconnecting these basic cells in various ways. • The idea is consciously to design the development or maintenance process to address the anticipated issues and problems. – Quality, product technology, unknown requirements, unstable requirements, misunderstood requirements, complexity

Product Technology Unknowns: • Hardware engineers have long known that new, more complex, or novel products rarely work as expected and are never directly shippable. – They thus build breadboards to test their technical concepts and experiment with alternative approaches. • With software it is also appropriate to breadboard critical, complex, or unusually demanding functions. • Experimental Kernel K_e • ETX Specifications for the Unil Implementation Kernel K_e

Problem of Complexity: The process kernels described thus far have focused on small tasks that could be performed by one or two programmers. • While these kernels provide useful guidance for small tasks, their real value is the insight they can provide for larger projects. • Here multiple modules are typically involved, and the work of many professionals must be coordinated. • Integration Cycle • Integration Cell • Build Cell

Requirements Instability: Assumed that the software requirements were known, stable, and understood. Since this is rarely the case, compensating process provisions are needed. • KSU' – The requirements are known and stable, but the implementers do not understand them sufficiently well to produce an adequate implementation (not U, or U'). • KS'U – The requirements are known and understood, but they are not stable (S'). • K'SU – The requirements are stable and understood, but they are not fully known (K').

Prototype Process Models: Experimental Kernel K3 (KSU') Requirements Not Understood • Requirements Unstable (KS'U) • Requirements Unknown (K'SU) • Prototype programs can be built to learn the potential customers' reactions. • These prototypes demonstrate one or more facets of system behavior for test with the intended users. • They can then be used to try to reduce the requirements uncertainties

Use of Prototyping: Each of the prototyping methods could be used at any of the process levels. • In a large system some modules may require some form of prototyping, while others could be developed directly. • With external interface, for example, the involved modules should often be prototyped and tested with the end users.

Prototyping Issues: Evidence shows that product development through prototyping is substantially faster and less expensive than traditional methods. – Even if it is only intended as a quick experiment, the prototype's objectives should be clearly established before starting to build it. – Define the prototype process in advance.

Detailed Process Models: All the process models in the world will not help to produce programs unless they can be reduced to the level of programming. • This is the atomic process level, which is the level at which the detail used in task definition should be appropriate to the knowledge and skill of the professionals. • One example is the partial decomposition of the process for building regression test buckets. – Regression Test Refinement– The regression test process starts with the regression test plan. • Regression Test Planning– Regression test planning is further decomposed.

Activity Cycle Models: The process models we have discussed so far are like state models of a software system. • In simplest terms, the process is either in an idle state before the entry criteria for the first cell are met, or the process is in a state represented by one of the succeeding cells. • After the final output from the last cell is produced, the model returns to the idle state, waiting for further requirements. • As the implementation tasks become more and more complex, the simplistic picture may provide a less realistic representation of actual task behavior. • One alternative is to consider basing process models on entities. – Here one deals with entities and the actions performed on them. – Each entity is a real object that exists and has an extended lifetime. • Examples of entities – The deliverable code – The users' installation and operation manuals – The requirements documents – The design – The test cases and procedures • Entity process models (EPMs) provide a useful additional representation of the software process because they are often more accurate than task-based models for complex and dynamic processes. • Each entity cycles through a set of states before it will be complete as nearly involved, and the work of many professionals must be coordinated. • Integration Cycle • Integration Cell • Build Cell

Requirements Instability: Assumed that the software requirements were known, stable, and understood. Since this is rarely the case, compensating process provisions are needed. • KSU' – The requirements are known and stable, but the implementers do not understand them sufficiently well to produce an adequate implementation (not U, or U'). • KS'U – The requirements are known and understood, but they are not stable (S'). • K'SU – The requirements are stable and understood, but they are not fully known (K').

Prototype Process Models: Experimental Kernel K3 (KSU') Requirements Not Understood • Requirements Unstable (KS'U) • Requirements Unknown (K'SU) • Prototype programs can be built to learn the potential customers' reactions. • These prototypes demonstrate one or more facets of system behavior for test with the intended users. • They can then be used to try to reduce the requirements uncertainties

Use of Prototyping: Each of the prototyping methods could be used at any of the process levels. • In a large system some modules may require some form of prototyping, while others could be developed directly. • With external interface, for example, the involved modules should often be prototyped and tested with the end users.

Prototyping Issues: Evidence shows that product development through prototyping is substantially faster and less expensive than traditional methods. – Even if it is only intended as a quick experiment, the prototype's objectives should be clearly established before starting to build it. – Define the prototype process in advance.

Detailed Process Models: All the process models in the world will not help to produce programs unless they can be reduced to the level of programming. • This is the atomic process level, which is the level at which the detail used in task definition should be appropriate to the knowledge and skill of the professionals. • One example is the partial decomposition of the process for building regression test buckets. – Regression Test Refinement– The regression test process starts with the regression test plan. • Regression Test Planning– Regression test planning is further decomposed.

Activity Cycle Models: The process models we have discussed so far are like state models of a software system. • In simplest terms, the process is either in an idle state before the entry criteria for the first cell are met, or the process is in a state represented by one of the succeeding cells. • After the final output from the last cell is produced, the model returns to the idle state, waiting for further requirements. • As the implementation tasks become more and more complex, the simplistic picture may provide a less realistic representation of actual task behavior. • One alternative is to consider basing process models on entities. – Here one deals with entities and the actions performed on them. – Each entity is a real object that exists and has an extended lifetime. • Examples of entities – The deliverable code – The users' installation and operation manuals – The requirements documents – The design – The test cases and procedures • Entity process models (EPMs) provide a useful additional representation of the software process because they are often more accurate than task-based models for complex and dynamic processes. • Each entity cycles through a set of states before it will be complete as nearly involved, and the work of many professionals must be coordinated. • Integration Cycle • Integration Cell • Build Cell

Establishing and Using a Process Definition: • Each software organization should establish a process architecture and process models tailored to its particular needs. • This tailoring is done as follows: – Define a standard process as a foundation for tailoring. – Establish the ETX specifications for the standard process model. – Make provisions to gather and track the resulting process measurements. – Establish checkpoints and standards for SQA review. – Incorporate specific measurement and reporting provisions. – Instruct the development personnel on the use and value of the process architecture, the standard process models, and when, why, and how they should be tailored. – Since many projects will likely need their own unique process definitions, they should start with the standard process and then take the following steps: – Identify the unique project issues, problems, and success criteria. – Document the adjustments required to the standard process to produce a basic overall project process. – For each software system component, repeat these definitions. – Once each program module has been identified, create a process definition for it as well and make any necessary adjustment.

Basic Process Guidelines: For developing and using a process architecture and its process models: – Establish objectives for each project's process – Define the basic process architecture, make sure it meets the needs of the projects, and then enforce it as an overall process framework – Remember that each project, component, and module is unique, and its process should be uniquely determined. – Establish process definition standards. – Change the process model dynamically as the problem change. – Require that all deviations for the standard process be documented, reviewed, and approved.

CH 14. The Software Engineering Process Group

Software Engineering Process Group • In approaching the software process it is instructive to examine the way processes are handled in other fields. • While there are vast differences between software engineering and the typical factory, software process development also requires some specialists to do the work. • Changing anything as complex as a software process must necessarily involve a host of factors, among which one of the most challenging is the need to change the way the software people do their work. • Since change is difficult even without tight schedule pressure, it is not surprising that software process change is both painfully slow and haphazard. • Tasks for software process improvement: – Identify the key problems. – Establish priorities. – Define action plans. – Get professional and management agreement. – Assign people. – Provide training and guidance. – Launch implementation. – Track progress. – Fix the inevitable problems. • Software Engineering Process Group (SEPG) is the focal point for the total effort involved in changing the SW process. • Other people – Senior management – Line projects – SQA – Education, finance, and administration – Software professionals

Role of the SEPG • SEPG has two basic tasks that are done simultaneously: – Initiating and sustaining process change – Supporting normal operations • The SEPG may also serve as the spawning ground for other groups such as technology support, education, cost estimating, standards, and possibly even SQA. – Large software groups need professional support in all these areas, and, where it is not available, the SEPG can provide guidance until such functions are established. • While projects must not be disrupted with excessive change, the software process should be viewed as continuous learning. • Process change involves learning new methods and accommodating to the changing nature and scale of the problems encountered. • Change is also need to bring current software practice up to the level of current knowledge. – We typically know how to do our work far better than we are currently doing. • Most software problems are caused not by lack of knowledge but by the lack of the discipline to apply the best-known methods. – Building the disciplined practices to do every software task with precise correctness requires a painstaking improvement program. • Changes are therefore a normal and continuous part of software management, and a key SEPG role is to ensure that these changes are effectively implemented. • The SEPG should also serve as a consolidating force for the changes that have already been made and should support the projects as they use new methods, standards, and technology.

SEPG as a Change Agent • A change agent provides the energy, enthusiasm, and direction needed to overcome resistance and cause change. • The SEPG fills this role by providing the skilled resources, the creativity, and the management leverage needed to make things happen. • The decision to make the changes must rest with line management, but the SEPG should provide the technical guidance on what changes are most important. • When management approves the action plans, the SEPG takes the lead in launching the required efforts: – Providing leadership in getting them staffed – Supporting the work with needed information, training, and skills • The SEPG also tracks action plan progress and informs management of status and major problems. • The continuing role of the SEPG can be divided into six categories: – Establish process standards. – Maintain the process database. – Serve as the focal point for technology insertion. – Provide key process education. – Provide project consultation. – Make periodic assessment and status reports.

Establishing Standards: • SEPG is the focal point for establishing process standards. • These include the product-related standards and the process-related standards discussed in the previous chapters. • Since the SEPG staff cannot have sufficient resources or the resident expertise to develop all these standards, it must establish working teams with members drawn from the most involved groups. • SEPG standards role involves the following tasks: – The SEPG recommends the priorities for developing standards. – The SEPG should ensure that prior work on the subject is reviewed. – The standards development team should include available experts and users. – The final standard review and approval is the most important step.

Process Database: Permanent repository for the data gathered on the SE process and the resulting products. • This data provides the reference for improving estimating accuracy, analyzing productivity, and assessing product quality. • It is used by process specialists, project professionals, the quality staff, and management. • The types of information retained in the process DB: – Size, cost, and schedule data – Product metrics – Process metrics • Considerations for an effective process DB: – The reason for gathering each piece of data must be stated. – To define the data, the exact meaning of each field must be specified, together will all the anticipated options and exceptions. – Simple and easy-to-use procedures are required for gathering the data, together with tools, forms, training, and assistance. – Means are also required for ensuring that the data is obtained in a timely way. – Means are needed to verify data accuracy. – Resources and procedures are required for entering the data into the DB. – Provision must be made for user access to and analysis of the data. – Provisions are required to protect and maintain the data.

Technology Insertion Focal Point: • Technology support for any reasonably large SE organization involves seven activities: – A process definition is needed to identify tasks and the set of requirements is developed for the needed support. – These requirements must be approved by the users, the people who will handle installation and support, and management. – A search is then made to identify commercially available tools that meet these needs. – An overall technology plan is developed. – A technology support group is established. – Education, training, and consultation must be provided.

Education and Training: • Education could also easily consume all the SEPG resources. • The SEPG must serve as the focal point for process education and avoid being saddled with all the teaching responsibilities. – Ideally a full-time education group maintains a staff of volunteers, instructors, or consultants to do this job. – Where an adequately staffed education group is not available, knowledgeable professionals can generally be found to serve as volunteer instructors. • Example courses: – Project management methods • How to plan, estimate, and track projects – Software design methods • The use of design languages, object-oriented design, and prototyping – Quality management • Methods for making quantitative quality estimates and plans – Design and code inspections

Process Consultation: • The key focus of the process group is on improving the practice of SE. • They do this by working with software practitioners both to provide assistance and to stay current on the project's problems. • If they do not stay current, they will not be able to relate to the practicing professionals. • Another reason for SEPG consultation is that few projects can afford their own SEPG group. • The SEPG can be most helpful to the project by consulting on: – The process data they should gather – The analyses and interpretation of the data gathered – Tuning the standard process to unique project needs – Assisting with the preparation of quality plans – Serving as experienced inspection moderators – Advising on the priority areas for technology insertion

Process Status and Assessment: • The SEPG is also responsible for understanding the current state of software practice and alerting management to key problems. – Awareness of how completely each project's process is defined – Knowledge of how the process is implemented – Judgment on when an assessment would be appropriate – Leadership in conducting assessments • The items required to make senior management reviews effective: – Goals for process improvement – A comparison of the actual process state to prior plans – The status of the process improvement actions – Identified problems and recommended corrective actions – Recommended responsibilities for handling these actions

Establishing the SEPG: • There are many issues to address in setting up an SEPG are: – How big should the effort be? – Where does the staff come from? – Who should head it? – Where should it report? – What are the tasks for initial focus? – What are the key risks? – How is its effectiveness evaluated?

SEPG Size and Staffing: • While each organization has its own unique needs, a useful initial guide is to aim at full-time assignments to the SEPG of about 2 percent of the software professionals. • For organizations of about a hundred or more software professionals, the typical range falls between 1 percent and 3 percent, with the prime determinants being the ability to recruit suitable candidates and the financial constraints of the organization.

SEPG Leadership and Reporting: • The SEPG leader must be a knowledgeable manager with a demonstrated ability to make things happen, the respect of the project professionals, and the support of top management. • To select official agents for process change: – Agents should be enthusiastic about leading the change process. – Agents must be both technically and politically capable of understanding the problems and ensuring that effective solutions are implemented. – Agents need the respect of the people they are to deal with. – Agents must have management backing for their activities or they will not act with the assurance needed to get wide cooperation and acceptance.

SEPG Priorities and Risks: SEPG priorities should roughly follow those given, with initial emphasis on project planning and project management. • The most important single guideline is for the SEPG to limit its focus to those tasks it can handle reasonably quickly and effectively.

Evaluating the SEPG: There are many ways to evaluate staff and support groups like the SEPG. • Some of the more common methods are to use advisory groups or periodically to poll the users and SW professionals for their opinions. • While it is probably a good idea to do some of these things, perhaps the best approach is to judge the SEPG on how effectively it applies the software process maturity framework to its own work. • *Level 2:* – Does the SEPG have a plan for its work, a tracking system, and means to retain and control its work products? • *Level 3:* – Have the SEPG professionals established a basic framework for their own work, including standards, procedures, and are view program? • *Level 4:* – Does the SEPG measure the productivity and quality of its own work? • *Level 5:* – Do they regularly assess their own activities for improvement opportunities and incorporate them in their working process.

CH 15. Data Gathering and Analysis

The Managed Process: The key steps required to advance from defined Process to Managed Process – Establish a minimum basic set of process measurement to identify the quality and cost parameters of each process step. – Establish a process DB and the resources to manage and maintain it. – Provide sufficient process resources to gather and maintain this process data and to advise project members on its use. – Assess the relative quality of each product and inform management where quality targets are not being met.

Data Gathering and Analysis: gather and analyze software data to help us improve the SE process. – Data gathering is expensive and time-consuming. – It affects the busiest people and may even be viewed as personally threatening. – There is also considerable confusion on what data to gather and how to use it.

The Principles of Data Gathering: The data is gathered in accordance with specific objectives and a plan. • The choice of data to be gathered is based on a model or hypothesis about the process being examined. • The data gathering process must consider the impact of data gathering on the entire organization. • The data gathering plan must have management support.

Objectives of Data Gathering 1. Understanding: As part of a research or development study, data can be gathered to learn about some item or process. 2. Evaluation: The data can be used to study some product (or activity) to see if it meets acceptance criteria. 3. Control: The data can be used to control some activity. 4. Prediction: The data can be used to develop rate or trend indicators.

Data Gathering Models • To measure the software process successfully, one must almost start by knowing the expected results. • With a poorly understood process, it is hard to be precise about the data that is needed. • After some study and evaluation, however, assumptions can usually be made about the key process events and their relationships. • Data is then gathered to verify these assumptions. • These measures are related to this conceptual model; the variations are examined, and changes are made to the process or the model to bring them into closer agreement.

Impact of Data Gathering on the Organization: – 1. effects of measurement on the people: When people know they are being observed, their performance will generally change. 2. Effects of the people on the measurements: It is essential to motivate the professionals to gather the needed data in a timely way.

Management Support: data gathering must be viewed as an investment- expensive and often considerable delay before benefits. Once useful results have been produced, however, the project managers generally will be more willing to support the work.

Data gathering process • issues of resources, cost, and accuracy- data gathering waste of time. * PM's reluctant: do not see the direct value to their projects, hesitant to insist on any long-term efforts that may affect immediate priorities. * Data gathering program must start with management support and be thoroughly planned and carefully validated.

Managing the Data Gathering Process: * Cost of Data Collection – Overhead to tasks (experiments) 3 – 7%: Forms, meetings, training, interviews, cost of using tools – Data processing 10 – 12%: Collecting/validating forms, archiving/entering data, data management and reporting – Analysis of information Up to 25%: Designing experiments, evaluating experiments, defining analysis tools. * Reason for expensive gathering and analyzing software data: manual work by people who are inexperienced at data gathering. The effort of recording time spent per task, results produced, or defects found is time consuming. With increasing experience and improved tools, however, these costs can be reduced. * Data gathering is expensive its process must be carefully planned and managed. The data must be precisely defined to ensure that the right information is obtained, and it must be validated to ensure that it accurately represents the process. * The data must be retained by someone who owns and maintains the process DB. * Summary of Data Gathering Problems: Problem, typical cause, and possible solutions • Data is not correct. • Data is not timely. • Data not measured or indexed properly • Too much data needed • Needed data does not exist.

Data Gathering Plan: Data gathering plan should be produced by the SEPG with the help of the projects and the participation of some of the professionals who will gather the data. – What data is needed, by whom, and for what purpose? – What are the data specifications? – Who will gather the data? – Who will support data gathering? – How will the data be gathered? – How will the data be validated? – How will the data be managed?

Data Validation: • Software data is highly error-prone and special validation provisions are generally needed. – The data must either be gathered automatically or extensively validated. • When people have been trained on the definitions and data gathering methods, periodic spot checks will generally suffice. • People who do this validation must be able to explain the data definitions, the data gathering methods, and the program's objectives.

Software measures: – Characteristics of several useful software measures – Examples of software data

Data Characteristics: • Software process measures following characteristics: – The measures should be robust. – The measures should suggest a norm. – The measures should relate to specific product or process properties. – The measures should suggest an improvement strategy. – They should be a natural result of the process. – The measures should be simple. – They should be both predictable and trackable. • Software measures can be classified in several ways: – Objective/subjective – Absolute/relative – Explicit/derived – Dynamic/static – Predictive/explanatory • The SE objective for data gathering and analysis to use an increasing number of objective, absolute, explicit, and dynamic measures to control and improve the way the work is done.

Software Size Measures: Size measures important because amount of effort required to do most tasks is directly related to the size of the program involved. – Unfortunately, there is no generally accepted measure of program size that meets all the previously described criteria. – There are no simple measures of software size because software size is not a simple subject.

Lines of Code: Major risk is that its unthinking use can motivate maximization of LOC counts. * The only solution is to use size measures as guides and not for evaluation of people or organization. * Alternative ways of counting LOC: – Executable lines – Executable lines plus data definitions – Executable lines, data definitions, and comments – Executable lines, data definitions, comments, and JCL – Physical lines on an input screen – Logical delimiters, such as semicolons

Normalized LOC Measures: Another problem with LOC counts is that with high-level languages they do not recognize the increased functional content of the source lines of codes. • Some organizations compensate for this by using factors to generate equivalent assembly language. • To be truly representative, such factors must reflect both the source and assembly languages being used, as well as the particular program functions involved.

Error Data:

Category	Item Measured	Causes
Errors	Human actions	Programmer mistakes
Defects	Program properties	Errors
Bugs	Program malfunctions	Program defects
Failures	System malfunction	Bug and other malfunctions
Problems	Human perceptions	Failure, human errors, human misconception

Process of Problem Analysis: • software error measurement problem is one of distinguishing between causes and effects. • Since our intent is both to correct and to prevent the errors that programmers make, we must work our way up this chain of possible measurable events.

Classes of Defect Measures: * software defects- major concern in both testing and operation –used as one okey process measurement. * SW defects categories: – severity, symptoms – where found, when found, how found – where caused, when caused, how caused – where fixed, when fixed, how fixed * Error Categories * Problem Discovery and Resolution Data * Frequency of Error Occurrence * Types of Design Errors * Types of Coding Error

Productivity Data: • How much effort is spent in each task of the software process? • For each product element, what effort was devoted to each process task? • What is the relative cost of defect removal by the various tests and inspections and by each major error category? • How effective are various methods and technologies in improving the productivity of each process task? • What is the resource history by major process task, and how should the planning factors be adjusted for subsequent projects?

Organizational Productivity: Productivity measures are most prone to misuse when they are applied at the highest project and organizational level. Desirable to have some measure of how the organization is performing over time. For software unfortunately, the only simple measures available are lines of code and effort expended.

Sensitivity of Productivity Measures: Software productivity numbers vary so widely among several companies as there are significant differences among the organizations' definitions of lines of code and labor actually charged to a project. * Productivity data can be highly misleading unless all the parameters are strictly defined and controlled.

Data Analysis: Data can be used to support software development and maintenance. As software organizations gather data and begin to use it, they will find many more ways to apply it.

Error Data Analysis: Exhaustive study of data analysis on the IBM DOS OS performed by Endres (1975) includes considerable detail on their error cause analysis results. – They concluded from this data that only about half of the mistakes can be avoided with better programming tools and techniques. – They found that an important benefit of gathering and analyzing such data is the focus it provides on the areas with the greatest likelihood of causing problems. • Questions to be Answered by IBM DOS Error Study • Number of Modules Affected by An Error – Three largest modules had the largest numbers of errors. • Number of Errors Per Module – The 21% of the modules that had more than one error accounted for 78% of the total errors. • Frequency of Errors – The changed code had a 60% higher error rate than new code (7.8 vs. 4.8).

Test Data Analysis: Many different analyses can be made of test data. – Typically, one can look at defect data rather than examining the test cases for programming defects. – Some example questions: • What is the relationship between test coverage by program module and the number of defects detected and remaining in the program? • Since test data can be obtained relatively easily, testing is one of the best places to start gathering and analyzing the SE process.

Analysis of Inspection Rates: • Wenneson describes using statistics for long- estimating, planning, and scheduling development and QA work. • Nominal Inspection Rates (FORTRAN design and code inspection) - Significant deviation from these values were cause for review of the way the inspections were conducted. • Errors Detected Versus Inspection Rate – The errors found per KLOC decline with increasing inspection rate. – While it is true that code with many errors is harder to review, this effect probably falls off at about 400 LOC/hour. – A rough upper limit of 0.3 to 0.4 KLOC/hr was probably about optimum for these FORTRAN programs. • Inspection Rate Versus Preparation Rate– Inspection preparation time should roughly equal review time. • Upper and Lower Ranges of Inspection Rates Versus Preparation Times – The upper and lower limits of inspection rates with various preparation times are plotted. – The LOC/hour of inspection plus preparation is thus between 60 and 100.

Inspection Data Analysis: Schulmeyer and McManus (1987) have provided some data on design and code inspections that can be analyzed with statistical techniques. – Selected Design Inspection Data • To facilitate analysis, the statistical control charts are next constructed.

Constructing Statistical Control Charts: • Inspection Control Chart – Lines of Code Inspected per Hour • Inspection Control Chart – Defects per Total Inspection Hours • Inspection Control Chart – Defects per Thousand Lines of Code • Inspection Control Chart – Hours of Preparation per Hour of Inspection • Percentage of Values Within Control Limits– Confidence interval • 68% within average +/- 1*SD • 95% within average +/- 2*SD – SD: standard deviation • Average of values)2 – (average of values)2]1/2 – UCL: average + 2*SD – LCL: average – 2*SD (0 if negative)

Using Control Charts: • The reason UCL and LCL values are important is that we need to identify unusual conditions and control them as we try to improve our processes. • It is thus important to screen out those values that are probably due to purely random variation and examine the rest.

Statistical Analysis of Test Data: Card (1987) has reported an example of how these same statistical techniques can be applied to test data. • Process Control Results

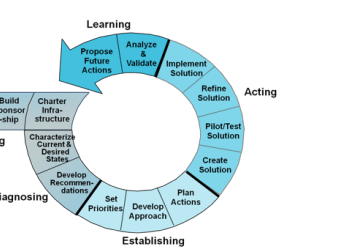
Other Considerations: Since it is easy to draw erroneous conclusions from poorly conducted measurements, it is important to remember some key points. – Since data can generally be interpreted in many ways, listen to what the data is saying rather than using it to reinforce preconceived opinions. – Provide resources to validate and analyze the data. – Make sure that process data is never used to evaluate people. – The simple act of gathering data will change the process. – Start carefully and don't attempt to do too much at one time. – Seek the views of the data collectors on the data collection process and any modifications, enhancements, or support they need. – Provide the initial results to the people the data concerns before giving it to anyone else. – Start with a comprehensive plan. – The debate on data definitions and formats can be endless

The IDEAL Model: Software Process Improvement Model

IDEAL℠ model: organizational improvement model that serves as a roadmap for initiating, planning, and implementing improvement actions. • Organizations that follow IDEAL approach to SW process improvement effectively integrate SEI technologies, courses, reviews of the way they do business in a comprehensive effort managing and improving their overall capacity. • The explosion of technological development has led organizations to adopt new technologies at an increasing rate. • The IDEAL Model provides an effective approach to adopting improved SE processes, methods, and tools.

Initiating: Laying the groundwork for a successful improvement effort -KEY • Business reasons, business goals, scope, … • CMMI – Continuous Representation (Capability Level) or Staged Representation (Maturity Level) – Body of Knowledge (SW Eng., Sys Eng. IPPD, Supplier Sourcing)

Error!



Diagnosing: Determining where you are relative to where you want to be - KEY • CMMI Appraisal (SCAMPI: Class A, B, C) • CL or ML • Process Areas (PAs) • Goals • Practices

Establishing: Planning the specifics of how you will reach your destination -KEY • PAs/Goals/Practices

Acting: Doing the work according to the plan- KEY • PAs/Goals/Practices

Learning: Learning from the experience and improving your ability to adopt new technologies in the future- KEY • CMMI Appraisal (SCAMPI: Class A, B, C) • Business reasons and goals

Mat. Level	Focus	Process Areas
5. Optimizing	Continuous process improvement	Organizational Performance Management (OPM) Causal Analysis and Resolution (CAR)
4. Quantitatively Managed	Quantitative management	Organizational Process Performance (OPP) Quantitative Project Management (QPM)
3. Defined	Process standardization	Requirements Development (RD), Technical Solution (TS), Product Integration (PI), Verification (VER), Validation (VAL), Organizational Process Focus (OPF), Organizational Process Definition (OPD), Organizational Training (OT), Integrated Project Management (IPM), Risk Management (RSKM), Decision Analysis and Resolution (DAR)
2. Managed	Basic project management	Requirement Management (REQM), Project Planning (PP), Project Monitoring and Control (PMC), Supplier Agreement Management (SAM), Measurement and Analysis (MA), Process and Product Quality Assurance (PPQA), Configuration Management (CM)

Interpretation Issues Encountered and How Addressed: • Artifacts – CMMI Typical Work Products • Terminology – Glossary • Training and Resources – Methods, tools, and technologies • Management issues – Support and resources – Estimation, plan, measurement, … – Process, process, process, … • SPI team

Key Activities That Were Necessary to Implement CMMI: • Identify the current practices (SCAMPI A/B/C) • Find problems and solutions (CMMI) • Implement a pilot solution • Evaluation • Implement all • Assessment (SCAMPI A/B/C) • Learn and iterate

Error!

Indicator Type	Description	Examples
Direct artifacts	The tangible outputs resulting directly from implementation of a specific or generic practice. An integral part of verifying practice implementation. May be explicitly stated or implied by the practice statement or associated informative material.	Typical work products listed in reference model practices Target products of an "Establish and Maintain" specific practice Documents, deliverable products, training materials, etc.
Indirect artifacts	Artifacts that are a consequence of performing a specific or generic practice or that substantiate its implementation, but which are not the purpose for which the practice is performed. This indicator type is especially useful when there may be doubts about whether the intent of the practice has been met (e.g., an artifact exists but there is no indication of where it came from, who worked to develop it, or how it is used).	Typical work products listed in reference model practices Meeting minutes, review results, status reports, presentations, etc. Performance measures
Affirmations	Oral or written statements confirming or supporting implementation (or lack of implementation) of a specific or generic practice. These statements are usually provided by the implementers of the practice and/or internal or external customers, but may also include other stakeholders (e.g., managers and suppliers).	Instruments Interviews Presentations, demonstrations, etc.